Genetic Architectures of Quantitative Variation in RNA Editing Pathways

Tongjun Gu,* Daniel M. Gatti,* Anuj Srivastava,* Elizabeth M. Snyder,* Narayanan Raghupathy,* Petr Simecek,* Karen L. Svenson,* Ivan Dotu,[†,‡] Jeffrey H. Chuang,[‡] Mark P. Keller,[§] Alan D. Attie,[§] Robert E. Braun,*[,1] and Gary A. Churchill*[,1]

*The Jackson Laboratory, Bar Harbor, Maine 04609, [†]Research Programme on Biomedical Informatics, Department of Experimental and Health Sciences, Universitat Pompeu Fabra, Hospital del Mar Medical Research Institute, Dr. Aiguader, 88 Barcelona, Spain, [‡]The Jackson Laboratory for Genomic Medicine, Farmington, Connecticut 06030, and [§]Department of Biochemistry, University of Wisconsin, Madison, Wisconsin 53706-1544

ABSTRACT RNA editing refers to post-transcriptional processes that alter the base sequence of RNA. Recently, hundreds of new RNA editing targets have been reported. However, the mechanisms that determine the specificity and degree of editing are not well understood. We examined quantitative variation of site-specific editing in a genetically diverse multiparent population, Diversity Outbred mice, and mapped polymorphic loci that alter editing ratios globally for C-to-U editing and at specific sites for A-to-I editing. An allelic series in the C-to-U editing enzyme *Apobec1* influences the editing efficiency of *Apob* and 58 additional C-to-U editing targets. We identified 49 A-to-I editing sites with polymorphisms in the edited transcript that alter editing efficiency. In contrast to the shared genetic control of C-to-U editing, most of the variable A-to-I editing sites were determined by local nucleotide polymorphisms in proximity to the editing site in the RNA secondary structure. Our results indicate that RNA editing is a quantitative trait subject to genetic variation and that evolutionary constraints have given rise to distinct genetic architectures in the two canonical types of RNA editing.

KEYWORDS genetics; RNA editing; Diversity Outbred; *Apobec1*; secondary structure; Multiparent Advanced Generation Inter-Cross (MAGIC); multiparental populations; MPP

RNA EDITING in mammals occurs through deamination of adenosine, which is converted to inosine (A-to-I editing), or deamination of cytosine, which is converted to uracil (C-to-U editing) (Davidson and Shelness 2000; Bass 2002). Other types of editing have been reported, but these findings remain controversial (Bass *et al.* 2012; Gu *et al.* 2012). The two canonical editing types, A-to-I and C-to-U editing, are mediated by distinct pathways. A-to-I editing is catalyzed on double-stranded (ds) RNA by proteins in the adenosine deaminase, RNA-specific (ADAR) family (ADAR1 and ADAR2) and is most common in neuronal tissues. However, the *Adar* gene family is ubiquitously expressed, and editing has been reported in many other tissues (Gu *et al.* 2012). Homozygous deletion of *Adar* genes is embryonic lethal in mice, and defects in A-to-I editing have been associated with neurodegenerative disorders and cancers (Gurevich *et al.* 2002; Paz *et al.* 2007). The C-to-U editing pathway is catalyzed by apolipoprotein B messenger RNA (mRNA) editing enzyme catalytic polypeptide 1 (*Apobec1*), which is expressed primarily in small intestine and liver, where it targets the transcript of apolipoprotein B (*Apob*), converting a CAA (glutamine) codon within the coding sequence to a stop codon (UAA). This editing event results in two APOB protein isoforms, APOB48 from the edited transcript and APOB100 from the unedited transcript. Editing of *Apob* is evolutionarily conserved and occurs in mice, humans, and other mammals. The edited isoform APOB48 functions in the synthesis, assembly, and secretion of chylomicrons in the small intestine; the unedited isoform APOB100 is expressed in the liver and gives rise to very low-density lipoprotein (VLDL), which is converted to LDL in the bloodstream. Although VLDL can contain either APOB48 or APOB100, LDL exclusively contains APOB100 (Davidson and Shelness 2000). Mice carrying a

doi: 10.1534/genetics.115.179481
Manuscript received June 15, 2015; accepted for publication November 17, 2015; published Early Online November 24, 2015.
Supporting information is available online at www.genetics.org/lookup/suppl/doi:10.1534/genetics.115.179481/-/DC1
[1]Corresponding authors: The Jackson Laboratory, 600 Main St., Bar Harbor, ME 04609. E-mail: Bob.Braun@jax.org and Gary.Churchill@jax.org

homozygous null allele of *Apobec1* are viable but exhibit abnormal lipid homeostasis (Hirano *et al.* 1996).

Both A-to-I and C-to-U editing alters RNA nucleotides at specific positions in a tissue-specific manner (Nakamuta *et al.* 1995; Bass 2002; Dawson *et al.* 2004). Recent reports have described editing events at tens to thousands of sites in humans and mice (Li *et al.* 2009; Rosenberg *et al.* 2011). Editing is often incomplete, with only a proportion of available transcripts being edited. The mechanisms that determine the specificity and efficiency of RNA editing are not well understood. Previous studies in humans have shown only limited effects of genetic variation on RNA editing. In one study, six A-to-I editing sites were found to be edited consistently across 32 individuals (Greenberger *et al.* 2010). Another study found evidence of genetic variation for two (of 7389) A-to-I editing sites (Daneck *et al.* 2012). Still another study found an association between RNA editing rates and genetic variation of *Apobec1* (Hassan *et al.* 2014). The broader impact of genetic variation on RNA editing in humans and mice remains unclear. Identification of allelic variants that alter the editing process could provide new insights into these mechanisms, and this provides the motivation for our genetic mapping study.

The Diversity Outbred (DO) population is a multiparent outbred mouse population derived from the same eight progenitor lines as the Collaborative Cross (CC) (Svenson *et al.* 2012). The DO population provides high levels of allelic diversity and high-resolution genetic mapping. Here we use natural allelic variants in the DO population to identify polymorphic loci that affect RNA editing with the aim of understanding the genetic factors that determine quantitative levels of editing. We consider the editing ratio—the proportion of edited reads at a site—as a quantitative trait for genetic analysis. Our findings reveal distinct modes of genetic regulation in the two editing pathways and provide insight into evolutionary constraints on the mechanisms that determine the specificity and efficiency of RNA editing.

Materials and Methods

DO mice

We obtained DO mice (J:DO, Stock #009376; 277 in total, 143 females and 134 males) from The Jackson Laboratory. Animals were received at 3 weeks of age and housed from wean age with free access to either standard rodent chow containing 6% fat by weight (73 females and 68 males; LabDiet 5K52, Scott Distributing, Hudson, NH) or high-fat chow containing 22% fat by weight (70 females and 66 males; TD.08811, Harlan Laboratories, Madison, WI). Animals were phenotyped for multiple metabolic and hematologic parameters as described by Svenson *et al.* (2012) and euthanized at 26 weeks of age. Liver samples were collected from each animal and stored in RNAlater solution (Life Technologies, Grand Island, NY) at −80°. All procedures on DO mice were approved by the Animal Care and Use Committee at The Jackson Laboratory (Protocol #06006).

DO founder strains

Breeder pairs for each of the eight DO founder strains, A/J, C57BL/6J, 129S1/SvImJ, NOD/ShiLtJ, NZO/HlLtJ, CAST/EiJ, PWK/PhJ, and WSB/EiJ, were purchased from The Jackson Laboratory and were bred to produce a total of 128 male mice, 16 per founder strain. Male progeny mice were maintained on standard breeder chow (Purina LabDiet 5013) until weaning at the University of Wisconsin–Madison. Beginning at 4 weeks of age, half the male mice from each strain were maintained on either a semipurified control diet containing 17% of kilocalories from fat (TD.08810) or a high-fat, high-sucrose (HF/HS) diet containing 45% of kilocalories from fat and 34% (by weight) sucrose (TD.08811); diets were from Harlan Laboratories (Madison, WI). Owing to reduced litter size or poor breeding, male CAST/EiJ and NZO/HlLtJ mice were purchased from The Jackson Laboratory at ∼3 weeks of age and switched to the control or HF/HS diet at 4 weeks. Animals were euthanized at 26 weeks of age (except for the NZO/HlLtJ mice). NZO/HlLtJ mice were euthanized at 20 weeks of age owing to high lethality in response to the HF/HS diet. Liver samples were collected, snap frozen, and shipped on dry ice to The Jackson Laboratory for RNA-seq analysis. All animal procedures were approved by the Animal Care and Use Committee at University of Wisconsin–Madison (Protocol #A00757-0-07-11).

RNA sequencing

Total liver RNA was isolated using the TRIzol Plus RNA Extraction Kit (Life Technologies) with on-column DNase digestion. Following the Illumina TruSeq standard protocol, indexed mRNA-seq libraries were generated from 1 μg total RNA followed by quality control and quantitation on an Agilent Bioanalyzer (Santa Clara, CA) and the KAPA Biosystems Library quantitative PCR (qPCR) quantitation method. Finally, 100-bp single-end reads were sequenced using the Illumina HiSeq 2000 (San Diego, CA). To minimize technical variation (lane and barcode effects in sequencing), the samples were randomly assigned to lanes and multiplexed at 12 or 24 samples per lane with randomly selected barcodes. Each DO sample was sequenced with two or four technical replicates to obtain ∼10 million reads per sample. Base calling was performed using CASAVA v1.8.0, and FASTQ files were filtered to remove low-quality reads using the Illumina CASAVA-1.8 FASTQ Filter (http://cancan.cshl.edu/labmembers/gordon/fastq_illumina_filter/).

RNA editing site prediction in founder strains

We constructed strain-specific transcriptomes for seven of the eight founder strains (except C57BL/6J) using ModTools (Huang *et al.* 2014). We incorporated strain-specific SNPs and short (<50 bp) indels from the Sanger Mouse Genomes Project (Keane *et al.* 2011; Yalcin *et al.* 2011) (REL-1303) into the Genome Reference Consortium Mouse Genome,

Build 38 (GRCm38) reference sequence. We constructed MOD files and strain-specific genomes and adjusted transcript annotation using the vcf2mod, insilico, and modmap utilities from the ModTools suite. We extracted the transcriptome of each founder strain using rsem-prepare-reference (RSEM v1.2.15) (Li and Dewey 2011) and built Bowtie indices (Bowtie v0.12.8) (Langmead *et al.* 2009). We searched for RNA editing sites using a two-tiered approach in which we searched for sites in the founder strains and then evaluated those sites in DO mice. A detailed description of the process with the script commands is available in Supporting Information, File S1 and Figure S1.

All 16 replicates from each strain were aligned to their respective strain-specific transcriptome using Bowtie with parameters -a –best –strata –v 3. We converted transcriptome alignments to genome alignments using rsem-tbam2gbam (Li and Dewey 2011) and adjusted the strain-specific coordinates to GRCm38 coordinates using Lapels (Huang *et al.* 2014).

We piled up the reads within each transcript (Ensembl archive 68) for each of the 16 replicates in a single DO founder strain. For each founder strain, we retained sites with the following properties:

1. Exactly two alleles expressed;
2. At least two reads in at least 75% of the 16 replicates;
3. Minor allele frequency (MAF) $>$ 5%;
4. Reference and edited allele coverage $\geq$ 160 reads;
5. No intersection with any known SNP or structural variant (Keane *et al.* 2011; Yalcin *et al.* 2012); and
6. Occurrence within a gene that contains no noncanonical editing sites.

Our reconstructions of founder strain and individual DO transcriptomes are based on the current mouse reference genome (GRCm38) and include annotated pseudogenes. However, some of the founder strain genomes are likely to harbor unannotated pseudogenes with paralogous SNP variants that, if they are transcribed, could appear to be editing. While we can account for known pseudogenes, unknown or unannotated pseudogenes are more challenging. The last filter (#6) was included to address these on the assumption that paralogous variants are likely to produce signatures of noncanonical editing. It is possible that some sites passing our filters are artifacts of expression of unannotated pseudogenes (File S2, File S3, File S4, and File S5). We have included counts of the number of canonical and noncanonical editing sites included throughout the process (Table S1).

We took the union of all editing sites from the eight founder strains and retained sites in genes with only canonical editing in all eight strains, coverage $\geq$ 160 at the edited bases, and less than 1% of reads in the nonedited bases. This produced a set of putative editing sites that we then queried and filtered in the DO samples (File S6).

RNA editing quantification in the DO samples

We built a combined transcriptome consisting of transcripts from all eight founder strains and indexed it using Bowtie (Langmead *et al.* 2009). We aligned RNA-seq reads from each DO animal to the combined transcriptome and separated them into individual alignments. We then converted the individual transcriptome alignments to genome alignments and then to GRCm38 coordinates using rsem-tbam2gbam and Lapels. Once all alignments were in the same coordinate system, we extracted only one instance of every read across all eight alignments and filtered any reads that were mapped to different coordinates in any two strains. This produced a BAM file for each DO sample representing alignments to all eight founder strains.

We piled up the reads in each DO sample at the putative editing sites and retained those with mean read depths $\geq$ 20 and a mean editing ratio $\geq$ 2%. We retained 186 sites (File S7).

Previous reports have noted that false-positive RNA editing sites may be due to a bias in the read position in which the putative editing site occurs (Pickrell *et al.* 2012). We looked at the position of each of the 186 editing sites in the 100-bp reads from each founder. We counted the frequency with which the editing site occurred at each position in the reads in each of the eight founders. We tested whether a greater proportion of reads occurred in the first and last 5 bp of each read than expected by chance using a binomial test (H_0: proportion = 0.1). We discarded 54 editing sites that occurred predominantly at the ends of reads and retained 102 sites (File S8).

We queried the RADAR (Ramaswami and Li 2014) and DARNED (Kiran *et al.* 2013) databases for all reported mouse editing sites and found 8891 sites. We piled these sites up in the DO samples and retained 98 sites with mean coverage $\geq$ 20 and a mean editing ratio $\geq$ 2% (File S9). Of these, 17 sites were identical to the *de novo* sites, and we retained 81 additional editing sites from RADAR and DARNED (File S9). We combined these with the 102 *de novo* sites and proceeded with a total of 183 sites.

Genotyping and haplotype reconstruction of DO genomes

Genotyping of the DO mice was described in our previous study (Svenson *et al.* 2012). DNA was extracted from tail biopsies and genotyped using the Mouse Universal Genotyping Array (MUGA; GeneSeek, Lincoln, NE). A total of 277 animals were genotyped. The founder haplotypes were reconstructed using a hidden Markov model based on the normalized intensity values from the MUGA (Gatti *et al.* 2014).

QTL mapping of the RNA editing ratio

The haplotype reconstruction process produced a matrix of eight founder allele dosages for each sample at each marker. We fit a linear mixed model by regressing the edited counts on sex, diet, and total counts with an adjustment for kinship between animals (Gatti *et al.* 2014). When total counts in a sample at an editing site were $<$10, we found that many false associations were produced by low total counts in the

denominator of the editing ratio. Therefore, we excluded samples with total counts <10 from the model because estimates of the editing ratio became unstable. The regression model used was

$$y_i = s_i\beta_s + d_i\beta_d + n_i\beta_r + \sum_{j=1}^{8} g_{ij}\beta_j + \gamma_i + \varepsilon_i \quad (1)$$

where y_i = edited counts for animal i; β_s = effect of sex; s_i = sex of animal i; β_d = effect of diet; d_i = diet for animal i; β_r = effect of total counts; n_i = total counts for animal i; β_j = effect of founder allele j; g_{ij} = founder allele dosage for founder j in animal i; and γ_i = random effect representing the polygenic influence of animal i. We report the LOD ratio as the mapping statistic. We determined significance thresholds by permuting the phenotype and covariate values 1000 times (Churchill and Doerge 1994, 2008). We selected the maximum association for each editing ratio and calculated the genome-wide p-value from the empirical distribution of null LOD scores. We applied a Benjamini and Hochberg false-discovery-rate (FDR) correction (Benjamini and Hochberg 1995) to these p-values and retained those with $p_{adj} < 0.05$ (File S10 and File S11).

Quantification of different isoforms of Apobec1

Two isoforms of *Apobec1* were found from the reference genome annotation. The difference between the two isoforms consisted of differences in the length of exon 4, one with a longer exon 4 (NM_031159) and one with a shorter exon 4 (NM_001134391). We quantified the expression of the two known isoforms and of a new isoform discovered in this study—the extension of exon 5. We constructed six isoforms of *Apobec1*, two of which were the reference isoforms. The remaining four isoforms were to test which of the two reference isoforms the aberrant extended isoform is derived from and the full length of the aberrant isoform. We embedded the six isoforms into the annotation file downloaded from Ensembl (http://useast.ensembl.org) and then used RSEM (Li and Dewey 2011) to quantify the six isoforms, tolerating zero mismatches in the alignments. Transcripts per million (TPM) and fragments per kilobase of transcript per million aligned reads (FPKM) were used for quantification and comparison.

Secondary-structure prediction

Full-length pre-mRNAs containing introns were imputed from strain-specific transcriptomes in which Sanger SNPs and indels were incorporated into the reference sequence. Minimum-free-energy structures were predicted for full-length mRNAs using the Vienna RNAfold tool (http://rna.tbi.univie.ac.at/cgi-bin/RNAfold.cgi) with default parameters (Lorenz *et al.* 2011).

We calculated the most stable substructure around the edit site using RNALfold (Hofacker *et al.* 2004) from the Vienna RNA Package v2.1.7 (Lorenz *et al.* 2011), which calculates the most stable substructures within a large sequence. By running RNALfold –z, we obtained all substructures with a Z-score ≤ -1.0 (lower Z-scores indicate more stable structures), from which we retrieved the substructure containing the edit site with the lowest Z-score.

We calculated the probability that the editing site occurs in a favorable editing position (occurring in a stem or a bulge or internal loop of size ≤ 2) by summing the pairing and unpairing probabilities of all relevant nucleotides around the editing site. For instance, the probability of the editing site being in a stem is the sum of all base-pairing probabilities between it and any other nucleotide (*i.e.*, the probability of the editing site being paired). For an editing site e to be in a bulge of size 1, the probability is that of the editing site being unpaired (1 − sum of all base-pairing probabilities for it) times the probability of nucleotide $e - 1$ being paired times the probability of nucleotide $e + 1$ being paired. An analogous calculation was performed for internal loops and larger sizes. File S13 provides the Python code for calculating the probability of the editing site being in a favorable position. Base-pairing probabilities were calculated by running RNAfold –p on the whole gene sequence.

Gene expression analysis

Gene expression was computed by summarizing the expression of all isoforms computed by RSEM (Li and Dewey 2011) from the founder strains. FPKM mapped reads were used. Similar expression results were obtained simply by summing the number of reads aligned to all the isoforms. The expression of APOB was computed by summarizing the reads that uniquely aligned to the *Apob* transcript from the eight founder strains.

Motif analysis

We used MEME (v4.10.0) (Bailey and Elkan 1994) to analyze the 30-nt downstream sequences adjacent to the 59 C-to-U editing sites that mapped to chromosome 6. We used the following settings: exactly one motif per sequence, motif length between 10 and 12 nt, and analysis of only the current strand. The described motif was the motif with the highest score. We searched for the motif in the same sequences using FIMO (v4.10.0) (Bailey and Elkan 1994), searching only on the given strand.

Genome assembly and annotation

We used genome coordinates from GRCm38 (Waterston *et al.* 2002). We obtained SNPs, indels, and structural variants for the eight DO founder strains from the Sanger Mouse Genomes Project v3 (Keane *et al.* 2011). We used the Ensembl archive 68 transcriptome (Flicek *et al.* 2012).

Data availability

The FASTQ files for the founder and DO RNA-seq runs are archived at the Gene Expression Omnibus under accession number GSE45684 (Munger *et al.* 2014). The alignment pipeline, including alignment and RNA editing site calling,

is described in File S1, File S2, File S3, File S4, File S5, File S6, File S7, File S8, File S9, File S10, File S11, and File S12.

Results

Our study employed liver RNA-seq data from two experiments. We profiled liver RNA in males from the eight inbred founder strains of the DO population. We also profiled liver samples in a genetic mapping panel of 277 DO mice of both sexes (Svenson *et al.* 2012). We implemented a conservative screen to identify candidate editing sites (see *Materials and Methods*). Our goal was not the exhaustive identification of editing sites. Rather, we aimed to obtain reliable editing sites and then to evaluate quantitative variation in editing. We selected sites with robust evidence for editing in the DO founders (editing $>$ 5% with at least 160 reads) and minimal evidence of sequencing or alignment errors, and we identified 192 sites in 131 genes. We retained 156 of those sites (51 A-to-I and 105 C-to-U sites) in 113 genes with mean coverage $>$ 20 and mean editing ratio $\geq$ 2% in DO mice. We removed sites that had an excess of editing sites in the proximal or distal 5 bp of each read and retained 102 sites. We then queried the RADAR (Ramaswami and Li 2014) and DARNED (Kiran *et al.* 2013) RNA editing site databases and found 98 sites with mean coverage $>$ 20 and mean editing ratio $\geq$ 2% for these sites in DO mice. Of these, 17 overlapped the 102 editing sites found in the *de novo* discovery pipeline. We retained a total of 183 sites (102 + 98 − 17) from the *de novo* discovery and the RADAR/DARNED databases (Table S2). There were 96 A-to-I sites and 87 C-to-U sites. Most of these sites (160) occur in the 3′ UTR of genes, two in intergenic regions, three in introns, three in noncoding RNAs or expressed pseudogenes, and 15 in coding regions. Among the 15 coding variants, 8 are synonymous, 6 are nonsynonymous, and 1, a well-known site in *Apob*, produces a stop codon (Chen *et al.* 1987). In earlier work, we had identified and confirmed 17 liver RNA editing sites (Gu *et al.* 2012); we detected 14 of these sites here, and the remaining 3 sites showed evidence of editing but were removed because they fell below our minimum read coverage criteria.

We mapped the editing ratio for each of the 183 sites and found significant associations (FDR $<$ 5%) for 119 sites in 81 genes. Of these sites, 70 were C-to-U sites, and 59 of these mapped to a region on chromosome *6*, while 11 mapped to a location near the edited site (Figure 1A and Table S1). The remaining 49 mapped associations were produced by A-to-I sites, and most of these mapped to a location near the edited site (Figure 1B).

C-to-U editing

C-to-U editing ratios for most sites were associated with a single region on chromosome *6*. We found 59 C-to-U editing sites with editing ratios that associated with a region of chromosome *6* between 119.6 and 125.6 Mb. Of these 59 sites, 50 came from the *de novo* pipeline, and 9 were from RADAR or DARNED. We examined the average C-to-U editing ratio in the DO population for these 59 sites and found that all but three sites in the Decorin (*Dcn*) gene were positively correlated with one another and shared a common pattern of allelic effects, suggesting that a single pleiotropic locus is driving variation in most C-to-U edited sites.

We averaged the editing ratios across all 59 positively correlated sites and mapped the average editing ratio to chromosome *6* (Figure 2A). The founder effects at the peak locus show a complex pattern (Figure 2B). DO mice carrying the CAST/EiJ, PWK/PhJ, and WSB/EiJ alleles have higher mean C-to-U editing; mice carrying the A/J, 129S1, and NOD/ShiLtJ alleles have intermediate editing; and mice carrying the C57BL/6J or NZO/HlLtJ alleles have low editing. The Bayesian credible interval for the association peak spans 2.5 Mb (121.1–123.6 Mb) and contains 39 transcripts, of which 10 are expressed in liver, *Apobec1*, *Phc1*, *Slc6a12*, *Slc6a13*, *M6pr*, *Mug1*, *Mug2*, *Necap1*, *Pex26*, and *Gm10319*. None of these genes has an obvious functional connection with RNA editing with the exception of *Apobec1* (apolipoprotein B mRNA editing enzyme catalytic polypeptide 1), the cytidine deaminase that catalyzes C-to-U editing (Petersen-Mahrt and Neuberger 2003). We hypothesize that *Apobec1* variants are the causal factor influencing quantitative variation in C-to-U editing.

To understand the effect of segregating alleles on the C-to-U editing ratio, we examined the canonical C-to-U editing site in *Apob*, which has an association on chromosome *6* at 120.2 Mb (Figure 3A). *Apob* transcript levels do not have an association on chromosome 6 in the liver, and thus, the editing association arises as a result of editing variation, not transcriptional variation. The pattern of allele effects for *Apob* C-to-U editing is similar to the pattern for the 70 C-to-U sites shown earlier (Figure 3B), suggesting that the underlying genetic mechanisms are similar. We imputed the founder SNPs onto the reconstructed DO haplotypes and performed association mapping within the support interval. We found that SNPs with alleles shared by C57BL/6J and NZO/HlLtJ contribute the highest LOD scores (Figure 3C). We estimated the editing ratio for each of the 36 genotypes present in the DO samples (Figure 3D). The genotype estimates are noisier owing to the smaller number of animals in each genotype class; nevertheless, there appears to be a continuous genotype-dependent variation of editing ratios, suggesting the presence of multiple functional alleles segregating in the DO samples. We examined the *Apob* editing ratios in the eight founder strains and compared them with the estimated additive effect of founder haplotypes obtained in the DO population (Figure 3, B and E). In both groups of animals, the C57BL/6J and NZO/HlLtJ haplotypes are associated with the lowest editing ratios, and the CAST/EiJ and PWK/PhJ haplotypes are associated with the highest editing ratios. Editing is somewhat elevated in the presence of the WSB/EiJ haplotype, while A/J, 129S1/SvlmJ, and NOD/ShiLtJ haplotypes are intermediate.

To better understand the mechanism driving the genetic variation in C-to-U editing, we examined SNPs and small insertions and deletions (indels) in *Apobec1* that distinguish

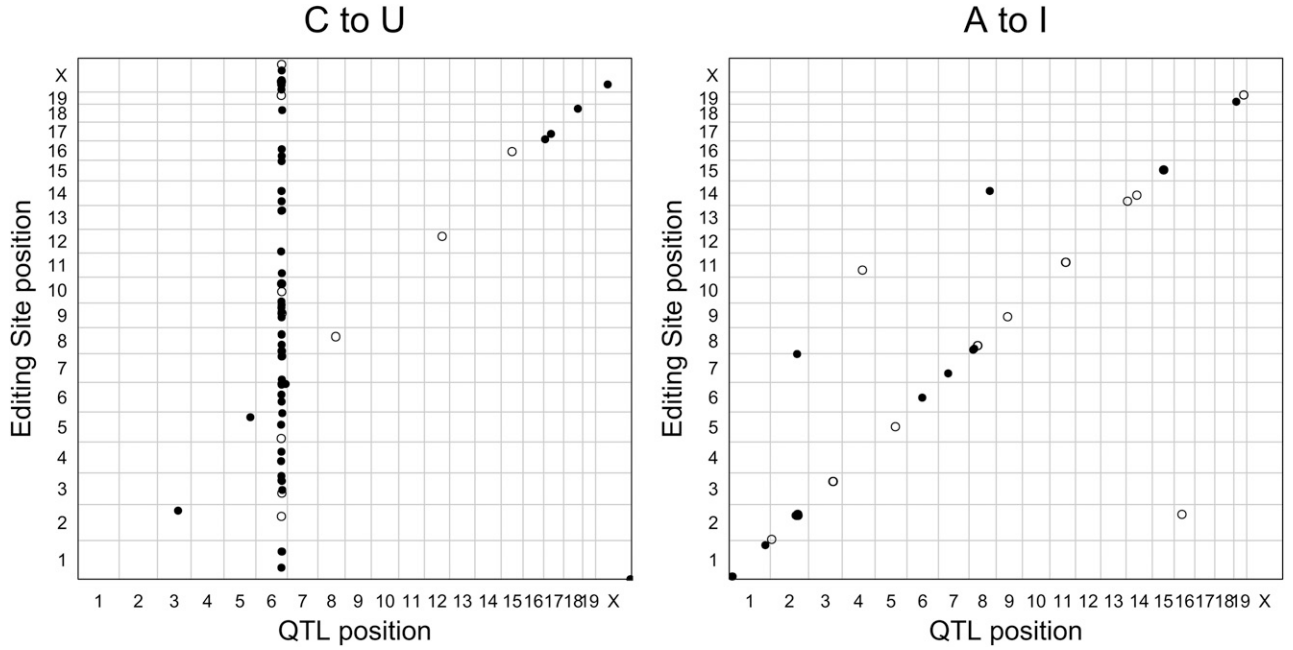

Figure 1 Mapping of C-to-U and A-to-I RNA editing reveals distinct patterns of genetic regulation. (A) Marker location (horizontal axis) *vs.* editing-site location (vertical axis) for the C-to-U editing sites shows that editing variation is due to genetic variants near the editing site (diagonal band) and at a location on chromosome 6 (vertical band). Open circles show sites from the RADAR or DARNED databases, and closed circles show sites from the *de novo* editing site discover. (B) Marker location *vs.* editing-site location for the A-to-I edit sites shows that editing variation is primarily due to genetic variants near the editing site. Note that some points represent multiple overlapping editing sites.

four haplotype groups: (1) CAST/EiJ and PWK/PhJ, (2) C57BL/6J and NZO/HlLtJ, (3) A/J, 129S1/SvlmJ, and NOD/ShiLtJ, and (4) WSB/EiJ. Fourteen SNPs or indels are shared by PWK/PhJ and CAST/EiJ, one of which creates an amino acid substitution in *Apobec1* (rs50844667, 122,581,478 bp); a positively charged arginine (R), located near the catalytic domain, is replaced by a neutral glutamine (Q) (Figure 4A). The glutamine variant found in CAST/EiJ and PWK/PhJ is shared by most species of mammals (Figure 4B). Only humans, rats (reference strain BN/SsNMcw), and most laboratory mouse strains, including the reference strain C57BL/6J, share the arginine residue at this site. The high editing ratio observed in CAST/EiJ and PWK/PhJ strains suggests that the glutamine allele of APOBEC1 may have increased catalytic activity.

Three *Apobec1* variants were shared by C57BL/6J and NZO/HlLtJ strains. One is a 262-nt insertion (relative to the other six strains) in the fifth intron (122,586,456–122,586,718 bp) (Figure 4C). Examination of RNA-seq reads at *Apobec1* in the eight founder samples shows an aberrant splicing pattern that occurs in ∼50% of transcripts from these strains; the fifth exon is extended by ∼1245 nt. This alternatively spliced transcript introduces 15 stop codons into the reading frame; thus, the mRNA may produce both a truncated transcript and a corresponding truncated protein. Overall, there appears to be a twofold reduction of total *Apobec1* expression in C57BL/6J and NZO/HlLtJ strains (Figure S2). However, when we estimated the normalized coverage of exon 6 alone, we found that the six strains that do not contain the intronic insertion have three- to fivefold higher coverage, suggesting that they may have three to five times the number of full-length *Apobec1* transcripts.

There is a single SNP shared by A/J, 129S1/SvlmJ, and NOD/ShiLtJ strains in exon 4 (rs51979390, 122,591,220 bp) (Figure 4C) that occurs at the splice junction and is associated with expression of a long *Apobec1* isoform (NM_031159) in these strains. The long isoform adds an extra 56 nt to the 5′ end of exon 4, which contains the 5′ UTR. The SNP is 40 nt downstream of the start of the longer exon 4 and 16 nt upstream of the shorter exon 4. Typically, the short isoform (NM_001134391) accounts for >95% of the total *Apobec1* mRNA (Nakamuta *et al.* 1995), but we observed high levels of the long isoform in A/J, 129S1/SvlmJ, and NOD/ShiLtJ strains. While expression of the long isoform is increased in these three strains, the total expression level of functional *Apobec1* is reduced compared to the WSB/EiJ strain. Expression of *Apobec1* is highest in WSB/EiJ (Figure S2). These polymorphisms define an allelic series at *Apobec1* with four functionally distinct groups. They combine to form 10 distinct genotypic classes in the outbred DO mice, which is consistent with the complex and continuous variation in C-to-U editing ratio (Figure 3D).

The “mooring” sequence, where the APOBEC1 complementation factor (A1CF) binds to *Apob*, is thought to be important in determining specificity of *Apob* editing (Smith *et al.* 2012). We searched for the mooring sequence within a 30-nt window downstream of the editing site in all 59 C-to-U editing sites with a strong association on chromosome 6 and

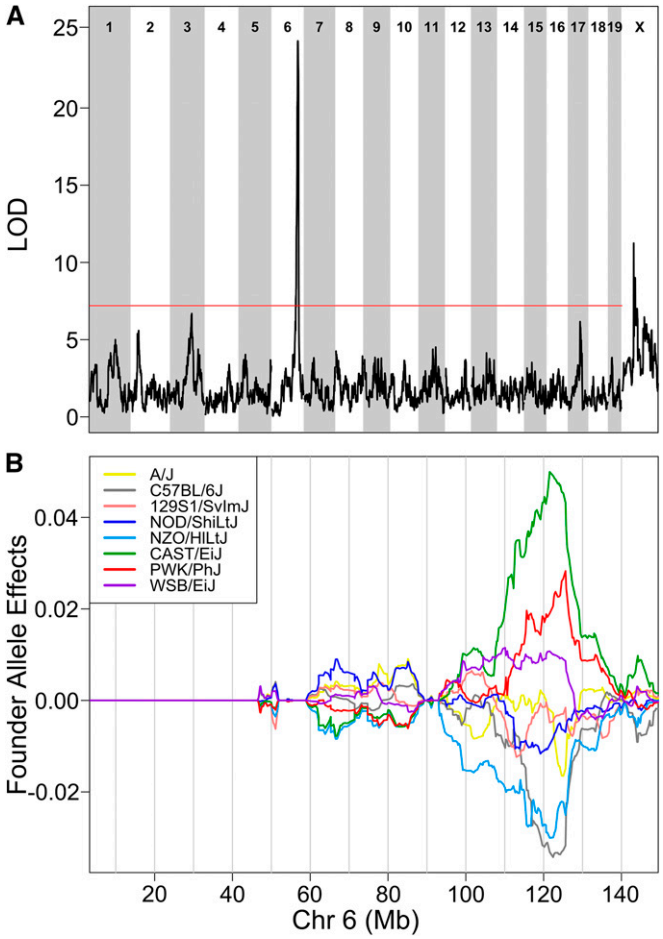

Figure 2 Mean C-to-U editing ratios for most editing sites map to a region on chromosome *6* at 122 Mb. (A) Genome scan of mean C-to-U editing for 70 editing sites shows a strong association on chromosome *6*. Horizontal axis shows the mouse genome; vertical axis plots the LOD score. Red line is the permutation-derived $\alpha = 0.05$ significance threshold. (B) Founder allele effects on chromosome *6* reveal a complex pattern of allele effects. Horizontal axis shows chromosome *6* in Mb. Vertical axis shows the founder allele effects.

found a consensus motif that is similar to the *Apob* sequence and is similar to previous reports (Figure 4D and Table S3) (Rosenberg *et al.* 2011). In *Apob*, the mooring sequence begins 3 nt downstream of the edited base. In other genes, the motif was found between 0 and 18 nt from the edited base. We did not find a correlation between the distance of the motif to the edited base and either the MEME enrichment *p*-value or the mean editing ratio in the DO population. Although we found SNPs in the mooring sequence of four sites and short indels (<50 nt) between the editing site and the distal end of the mooring sequence in two editing sites, we did not find any correlation between these genomic variants and the editing ratio. Thus, although *A1cf* may play a role in C-to-U editing, we did not find that genetic variation within the mooring sequence influences editing efficiency. Consistent with previous work, we observed enrichment of A and U bases adjacent to the edited base (Figure 4E) (Rosenberg *et al.* 2011). We also found that the bases adjacent to the editing site were enriched for sequences with an A in the 5′ position (χ^2 test, $p = 1.9 \times 10^{-4}$); 44 sites (75%) contained an A, whereas 15 sites (25%) contained a U.

Our results suggest that variation in C-to-U editing ratios in the DO population is regulated by an allelic series of the editing enzyme *Apobec1*. These findings are consistent with a recent report of editing variation in the AXB/BXA recombinant inbred panel (Hassan *et al.* 2014). Of the remaining C-to-U editing sites, 11 show local genetic associations that map to the region of the edited gene. Two C-to-U editing sites map to distant loci other than chromosome *6*. Six of the 11 sites have unusually high LOD scores (>30), and it is possible that some of these sites are the result of errors in the data or to unannotated paralogous variation. Of the local QTL, three genes with moderate but significant LOD scores, *Aldh6a1*, *Gramd1c*, and *Lamp2*, showed some evidence of regulation by the *Apobec1* locus. We searched for other distant QTL that might regulate C-to-U editing but did not find a consistent signal around the 11 local C-to-U editing sites. Curiously, the three sites in *Dcn* are anticorrelated with the pattern of allele effects at the *Apobec1* locus, an observation for which we currently have no explanation.

A-to-I editing

A very different picture emerges from genetic analysis of the 96 A-to-I edited sites detected in this study. Although approximately half (47) of these sites showed no evidence of editing ratio variation across the founder strains, a total of 49 A-to-I editing sites in 22 genes varied significantly across the founder strains and revealed significant associations in the DO population. Unlike C-to-U editing, where most of the sites displayed a similar pattern of allele effects, A-to-I editing sites were highly variable, suggesting independent genetic regulation for each site. The only exceptions were for editing sites located within the same gene where the pattern of allele dependence was similar for all sites within that gene.

These observations suggest that local genetic factors drive variation in A-to-I editing frequency. When we examined the support interval for each association, we found that the interval included the edited gene itself. It is known that the secondary structure of target mRNAs is an important factor for A-to-I editing site specificity (Dawson *et al.* 2004; Rieder and Reenan 2012). Thus, we hypothesized that *cis*-acting variants alter mRNA secondary structure near the editing site. To test this hypothesis, we examined the relationship between genetic variants, target RNA structure, and the edited sites. To do so, full-length target RNAs of each founder strain were imputed from known SNPs and indels. These RNAs were folded *in silico*, and their structure was examined for strain-dependent differences.

Because A-to-I editing enzymes prefer double-stranded regions of RNA (Lehmann and Bass 1999), we focused on the impact of genetic variants on double-stranded regions of the target mRNA and whether these regions were associated with edited sites. The long, noncoding (linc) RNA *0610005C13Rik* contains one editing site for which the

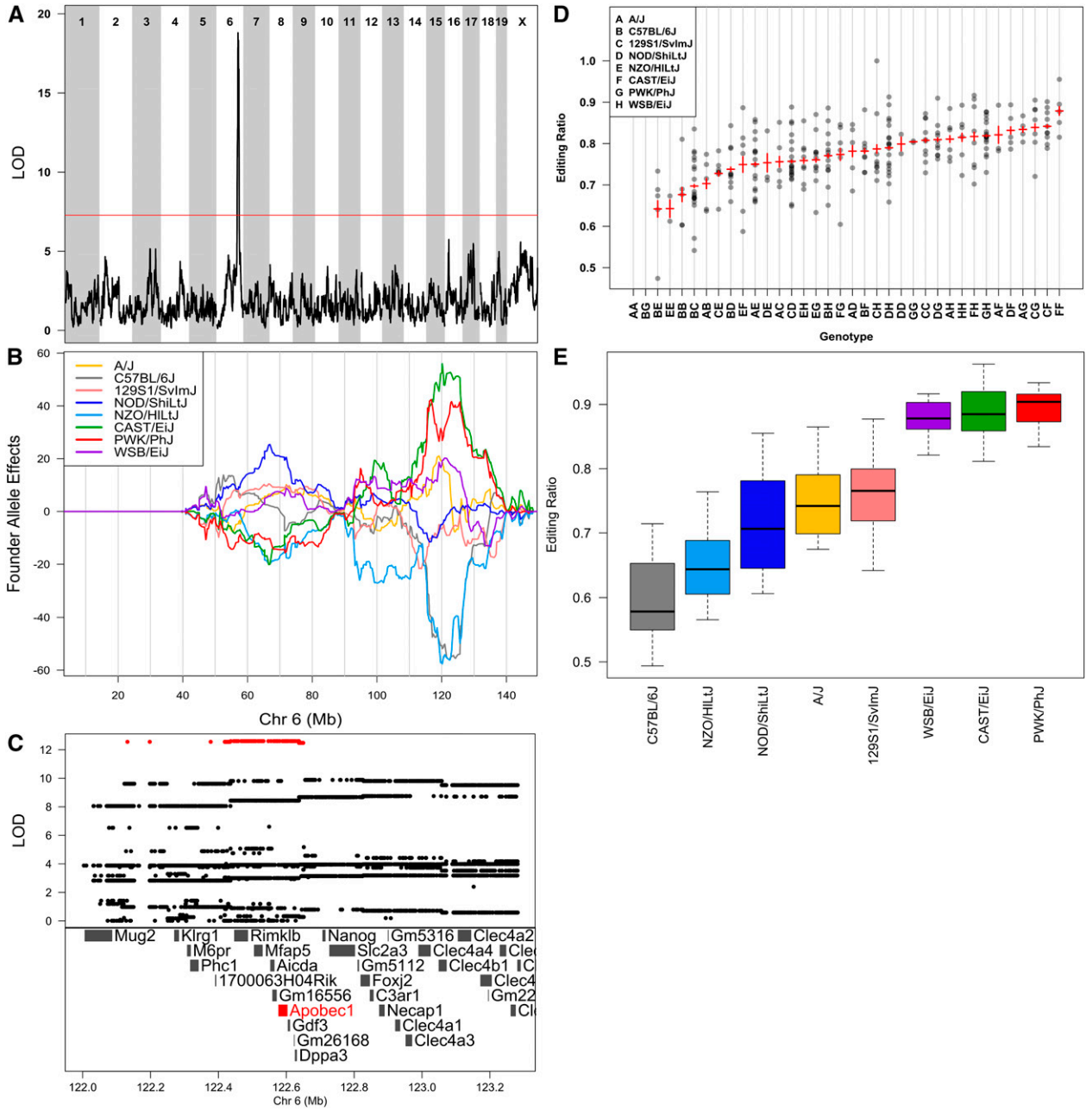

Figure 3 Genome scan of *Apob* C-to-U editing shows that editing is regulated by genetic variants on chromosome *6* that overlap *Apobec1*. (A) Genome scan of *Apob* editing shows a strong association on chromosome *6*. Horizontal axis shows the mouse genome; vertical axis shows the LOD score. Red line is the permutation-derived $\alpha = 0.05$ significance threshold. (B) Founder allele effect plots for *Apob* C-to-U editing on chromosome *6*. Horizontal axis plots Mb along chromosome *6*; vertical axis plots the founder allele effects from the linkage mapping model. Each colored line represents the effect of one of the eight founder alleles along the genome. (C) Association mapping within the QTL support interval shows a band of SNPs with high LOD scores (in red) that cover *Rimklb*, *Mfap5*, *Aicda*, *Gm16556*, *Apobec1*, *Gdf3*, *Gm26168*, and *Dppa3*. *Apobec1* is highlighted in red because it has been associated previously with *Apob* C-to-U editing. (D) Genotypes of DO mice at the location of the maximum LOD score (horizontal axis) *vs.* the editing ratio (vertical axis). DO mice may have one of 36 unphased genotypes, and these are denoted by two letters, each referring to one founder, along the horizontal axis. (E) *Apob* C-to-U editing ratio in the eight DO founders shows that the editing pattern is similar to the founder allele effects near *Apobec1*.

NZO/HlLtJ allele shows the highest editing (Figure 5, A and B). The NZO/HlLtJ allele carries 56 SNPs and 9 structural variants in *0610005C13Rik*, and these lead to a structural change compared to the reference sequence. In allele C57BL/6J, there is a 45-nt region of predominantly dsRNA around the editing site, while the NZO/HlLtJ transcript contains a 120-nt dsRNA region. RNAfold simulations of the ensemble of structures suggest that the editing site has a high probability of occurring in the stemlike region in all strains. However, the double-stranded feature is unusually stable in

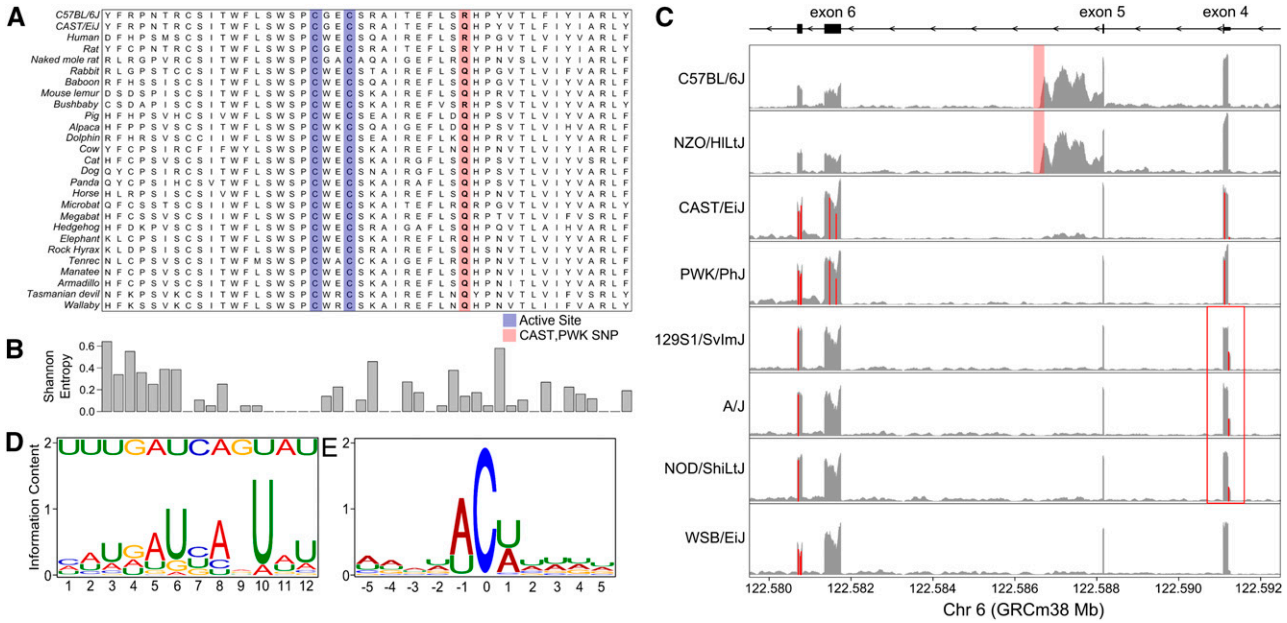

Figure 4 Genomic variants in the DO founders fit the pattern of *Apob* editing. (A) A nonsynonymous SNP in exon 6 of *Apobec1* contributed by CAST/EiJ and PWK/PhJ converts an arginine (R) residue to a glutamine (Q), highlighted in red. Residues in the active site are highlighted in blue. (B) Most mammals have a glutamine residue at this location. Shannon entropy for each base position shows that the glutamine is somewhat conserved. (C) RNA-seq pileups of *Apobec1* in the eight DO founders show transcriptional variation between alleles. C57BL/6J and NZO/HlLtJ alleles carry an insertion (shaded in red) in the fifth intron that overlaps a retained intron. 129S1/SvImJ, A/J, and NOD/ShiLtJ alleles carry a SNP (red box) in the 5′ UTR of exon 4 that has increased expression in those strains. The *y*-axis shows the read depth normalized to library size. (D) Consensus mooring sequence motif for C-to-U edited genes. (Top) The A*pobec1* mooring sequence. (Bottom) The consensus binding motif discovered using MEME. (E) Sequence information content around the edited C-to-U site shows that bases at the proximal and distal positions are either A or U. There does not appear to be sequence preference at other nearby base positions.

the NZO/HlLtJ strain, which shows the highest editing ratio ($Z = -13.77$ for the most stable locally folded structure of NZO/HILtJ). Given these observations and the fact that editing efficiency is known to positively correlate with dsRNA length, we examined other A-to-I editing sites and asked whether the editing site resided within regions of dsRNA that varied across strains.

Lactamase β2 (*Lactb2*) contains nine edited sites, of which five have significant QTL. The editing frequency at these five sites is low compared to other strains when the PWK/PhJ allele is present in DO mice (Figure 5D), and this effect is recapitulated in the founder strains (Figure 5E). The computed structure of the full mRNA of PWK/PhJ contains a shorter length of dsRNA than the reference C57BL/6J strain (Figure 5F), and this may reduce editing efficiency. Analogous to *0610005C13Rik*, the local stability of the *Lact2b* RNA structure trended with the editing ratios across strains. For example, the most stable local structure of PWK/PhJ is the least stable compared to the other strains, and PWK/PhJ has the lowest editing ratio. The gene Signal Peptide Peptidase Like 2A (*Sppl2a*) contains five A-to-I edited sites, and all have significant QTL. The founder allele effects in the DO population (Figure 5G) suggest that NOD/ShiLtJ alleles will have higher editing efficiency than CAST/EiJ alleles. Editing ratios in the founders (Figure 5H) are somewhat similar to the DO allele effects. NOD/ShiLtJ, which has higher editing than CAST/EiJ, has a longer stretch of dsRNA in the full mRNA folded structure than CAST/EiJ (Figure 5I). Recent reports have shown evidence of *cis*-acting intronic dsRNA elements that influence editing (Daniel *et al.* 2012). We searched for promiscuous editing in the three genes shown in Figure 5 but did not find evidence of editing at sites other than the target editing sites.

There were four distant associations among the A-to-I editing sites, and each association occurred in a different genomic location, suggesting that unlike C-to-U editing, there is no single gene that drives A-to-I editing in the livers of DO mice. There are some intriguing candidate genes under these associations. The association on chromosome *1* for *Tmem245* contains transfer RNA (tRNA) splicing endonuclease 15 homolog (*Tsen15*). The association on chromosome *2* for *Cd200r3* contains adenosine deaminase like (*Adal*). While we have no molecular evidence of distant regulation of A-to-I editing, these associations may represent interactions between edited transcripts and genes under the associations.

These observations demonstrate that variation of the A-to-I editing ratio in the DO population is driven primarily by local variants. Given the strain-dependent changes in target RNA structure, it appears that these variants play a role in modulating the local RNA structure of edited nucleotides.

Discussion

We used RNA editing ratio as a quantitative trait and found that most C-to-U editing sites in our study were controlled by a

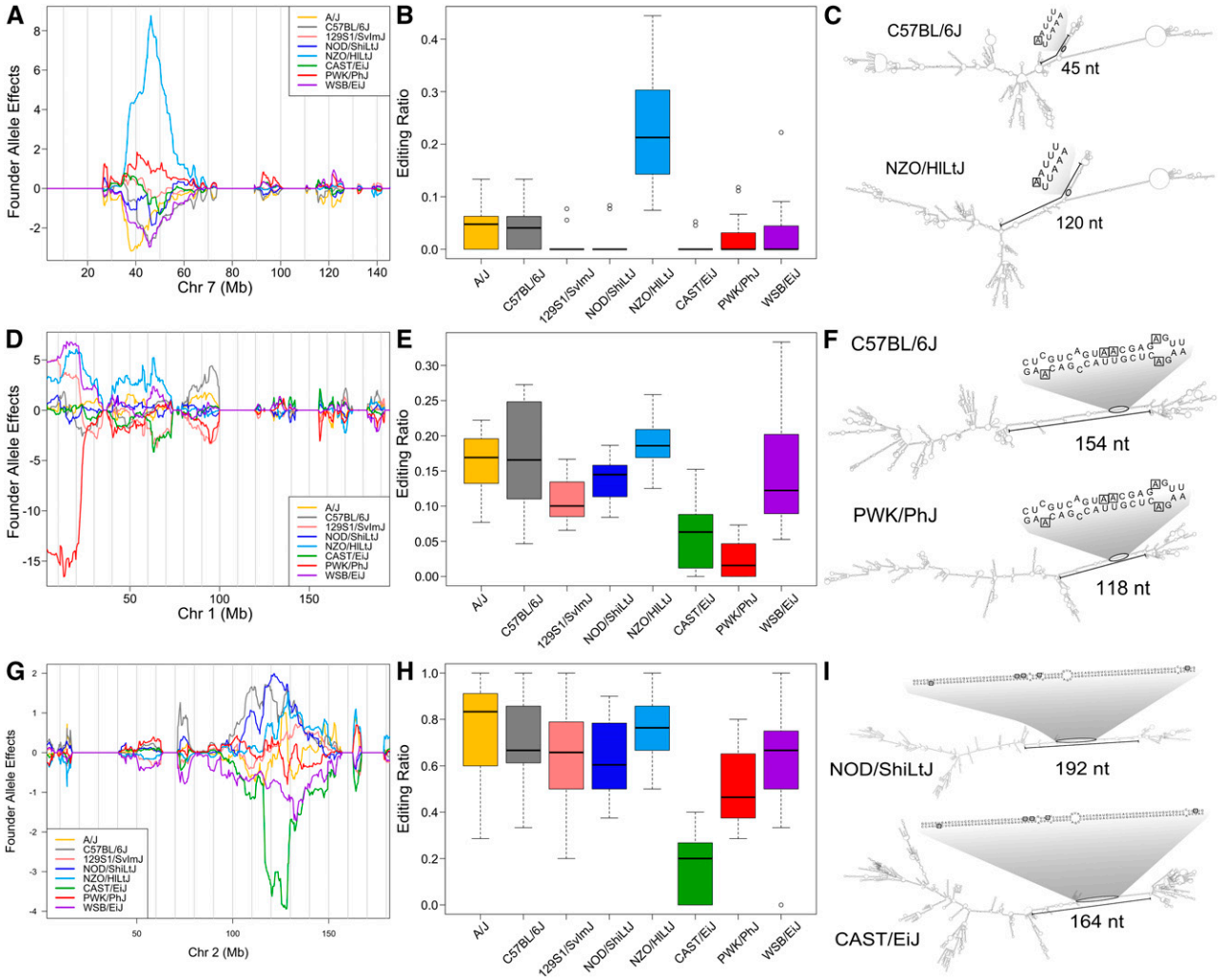

Figure 5 Structural differences between the founder strains may influence A-to-I editing efficiency. (A) Founder allele effects for *0610005C13Rik* show that DO mice carrying the NZO/HlLtJ allele have more edited transcripts. (B) Among the DO founder strains, NZO/HlLtJ has the highest editing ratio. (C) RNA secondary structure of the full mRNA for *0610005C13Rik* for C57BL/6J and NZO/HlLtJ shows that the NZO/HlLtJ strain contains a longer dsRNA region around the editing sites. The location of the editing sites on the full-length RNA is circled; the nucleotides around the editing site are enlarged and outlined by rectangles. (D) Founder allele effects for *Lact2b* show that DO mice carrying the PWK/PhJ allele have lower editing. (E) Among the DO founder strains, PWK/PhJ has the lowest editing ratio. (F) Full-length mRNA for *Lact2b* in C57BL/6J and PWK/PhJ shows that PWK/PhJ carries a shorter dsRNA region near the editing sites. (G) Founder allele effects for *2010106G01Rik* in DO mice show a complex pattern of editing ratios, with NOD/ShiLtJ having high editing and CAST/EiJ having low editing. (H) Editing ratios. The founders are similar to the allele effects observed in the DO population but differ in the order of editing ratios. (I) Full-length mRNA for *2010106G01Rik* shows that the dsRNA region around the editing sites is slightly shorter in CAST/EiJ.

single *trans*-acting factor at *Apboec1* that encodes an enzyme (APOBEC1) that catalyzes C-to-U editing. It is generally thought that APOBEC1 is positioned at the editing site by a mooring sequence (Smith *et al.* 2012) that is recognized by an APOBEC1 cofactor, A1CF. Consistent with this model, most of our C-to-U sites contain a mooring-like motif. However, we found no evidence for linkage at the *A1cf* locus itself, or linkage to variants within the mooring sequence, suggesting that genetic variation within the mooring sequences or *A1cf* does not drive variation in C-to-U editing in the DO population. The DO founder strains carry only nonsynonymous variants in *A1cf*, and there is little variation in transcript levels in the founder strains (Figure S2). Another RNA binding protein, RBM47, has been identified recently as a required APOBEC1 cofactor for C-to-U editing *in vivo* (Fossat *et al.* 2015). However, as with *A1cf*, we did not find evidence for linkage to *Rbm47* for any of the C-to-U sites studied, suggesting that neither putative cofactor of APOBEC1 is a driver of editing variation in the DO population. Our analysis strongly supports a role for *Apobec1* in determining quantitative variations in C-to-U editing. The C57BL/6J and NZO/HlLtJ *Apobec1* alleles with an alternative mRNA isoform are associated with both lower total mRNA and lower C-to-U editing. This outcome is particularly relevant given that most studies of C-to-U editing in mice use C57BL/6J, which may not be representative of murine C-to-U editing.

Apob editing is the most well-studied C-to-U editing event. Editing produces one of two versions of APOB, APOB48 (edited) and APOB100. APOB48 is involved in the transport of dietary fat by chylomicrons. APOB100 contains the LDL receptor binding site and is essential for the efficient clearance of LDL from the circulation. High levels of APOB100 are associated with atherosclerosis (Davidson and Shelness 2000). Given the large effect of polymorphisms on editing ratios in the DO population, it will be important to characterize how human genetic variation in *APOBEC1* affects C-to-U editing ratios and downstream phenotypes. There are 22 SNPs within the exon boundaries of human *APOBEC1*, and these may alter the APOB48-to-APOB100 ratio as well as circulating LDL/high-density lipoprotein (HDL) ratios.

In contrast to C-to-U editing, A-to-I editing in the liver seems to be regulated mostly by local factors. We found associations for 49 A-to-I editing sites, all of them in the vicinity of the edited gene. In many cases, these genetic variants appear to affect dsRNA stability and length in the region containing the edited sites of target RNAs. The major ubiquitous A-to-I editing enzyme, ADAR, shows distinct and strong preferences for long, stable dsRNA (Herbert and Rich 2001). Thus, the observed alterations in RNA structure are consistent with a model in which variant-induced dsRNA destabilization or length reduction negatively affects RNA editing efficiency. These findings are in good agreement with studies of natural genetic variation in *Drosophila* showing that local variants near the edited site are an important determinant of editing efficiency (Sapiro *et al.* 2015). While the *Adar* family of genes contain missense polymorphisms in the DO population, these variants do not appear to influence A-to-I editing in the liver.

ADAR-mediated editing occurs in many tissues, and the alteration of ADARs is known to be detrimental. We found little evidence for *trans*-acting associations despite the existence of six polymorphisms between the eight founder strains that alter amino acids in *Adar1* and several large indels in introns of *Adarb1*. While these polymorphisms may affect other non-editing-related functions of ADARs (Ota *et al.* 2013), these studies show that they have little or no impact on A-to-I editing in the liver. Our findings do not preclude the possibility that editing in other tissues may be affected by ADAR polymorphisms or by other proteins. In mammals, most A-to-I editing occurs in the brain, and in *Drosophila*, the fragile *X* mental retardation protein (FMRP) has been shown to affect editing at distant sites (Bhogal *et al.* 2011). In future studies it will be important to use natural genetic variation in the mouse to assess long- and short-range influences on RNA editing in the brain.

We have successfully identified associations and specific polymorphic loci that influence RNA editing. Both C-to-U and A-to-I editing is variable with clear underlying genetic drivers. We mapped multiple local associations for A-to-I editing sites and a shared distant association in *Apobec1* for C-to-U editing sites in the DO population. Most of the local A-to-I editing associations map to SNPs in the edited transcript, and we have identified candidate polymorphisms for some genes that are likely to affect mRNA secondary structure. The distinct genetic architectures discovered for C-to-U and A-to-I editing may reflect the different functional constraints in these two editing pathways. C-to-U editing of the transcriptome occurs at low ratios at sites other than *Apob*, and functional polymorphisms in the central catalytic enzyme appear to be tolerated. In contrast, A-to-I editing is widespread, and polymorphisms in central enzymes (i.e. ADARs) are more likely to have pleiotropic effects that are detrimental to the organism. Thus, when variation in A-to-I editing is present, the causal polymorphisms are local, limiting the effects to a single transcript.

Acknowledgments

This work was funded by National Institutes of Health (NIH) grants P50-GM076468 and R01-GM070683 to G.A.C. and National Cancer Institute grant CA34196 to The Jackson Laboratory in support of The Jackson Laboratory's shared services. J.H.C. was supported by NIH grant R21-HG007554. E.S. was supported by NIH grant K99/R00-K99HD083521. M.K. and A.A. were supported by NIH grants R01-DK066369, R01-DK058037, and R24-DK091207.

Literature Cited

Bailey, T. L., and C. Elkan, 1994 Fitting a mixture model by expectation maximization to discover motifs in biopolymers. Proc. Int. Conf. Intell. Syst. Mol. Biol. 2: 28–36.

Bass, B. L., 2002 RNA editing by adenosine deaminases that act on RNA. Annu. Rev. Biochem. 71: 817–846.

Bass, B., H. Hundley, J. B. Li, Z. Peng, J. Pickrell *et al.*, 2012 The difficult calls in RNA editing. Nat. Biotechnol. 30: 1207–1209.

Benjamini, Y., and Y. Hochberg, 1995 Controlling the false discovery rate: a practical and powerful approach to multiple testing. J. R. Stat. Soc. B 57: 289–300.

Bhogal, B., J. E. Jepson, Y. A. Savva, A. S. Pepper, R. A. Reenan *et al.*, 2011 Modulation of dADAR-dependent RNA editing by the Drosophila fragile X mental retardation protein. Nat. Neurosci. 14: 1517–1524.

Chen, S. H., G. Habib, C. Y. Yang, Z. W. Gu, B. R. Lee *et al.*, 1987 Apolipoprotein B-48 is the product of a messenger RNA with an organ-specific in-frame stop codon. Science 238: 363–366.

Churchill, G. A., and R. W. Doerge, 1994 Empirical threshold values for quantitative trait mapping. Genetics 138: 963–971.

Churchill, G. A., and R. W. Doerge, 2008 Naive application of permutation testing leads to inflated type I error rates. Genetics 178: 609–610.

Daneck, P., C. Nellåker, E. R., J. E. McIntyre, S. Buendia-Buendia, Bumpstead *et al.*, 2012 High levels of RNA-editing site conservation amongst 15 laboratory mouse strains. Genome Biol. 13: r26.

Daniel, C., M. T. Veno, Y. Ekdahl, J. Kjems, and M. Ohman, 2012 A distant *cis*-acting intronic element induces site-selective RNA editing. Nucleic Acids Res. 40: 9876–9886.

Davidson, N. O., and G. S. Shelness, 2000 APOLIPOPROTEIN B: mRNA editing, lipoprotein assembly, and presecretory degradation. Annu. Rev. Nutr. 20: 169–193.

Dawson, T. R., C. L. Sansam, and R. B. Emeson, 2004 Structure and sequence determinants required for the RNA editing of ADAR2 substrates. J. Biol. Chem. 279: 4941–4951.

Flicek, P., M. R. Amode, D. Barrell, K. Beal, S. Brent *et al.*, 2012 Ensembl 2012. Nucleic Acids Res. 40: D84–D90.

Fossat, N., K. Tourle, T. Radziewic, D. Liebhold, J.B. Studdert *et al.*, 2014 C to U RNA editing mediated by APOBEC1 requires RNA-binding protein RBM47. EMBO Rep. 15: 903–910.

Gatti, D. M., K. L. Svenson, A. Shabalin, L. Y. Wu, W. Valdar *et al.*, 2014 Quantitative trait locus mapping methods for diversity outbred mice. G3 4: 1623–1633.

Greenberger, S., E. Y. Levanon, N. Paz-Yaacov, A. Barzilai, M. Safran *et al.*, 2010 Consistent levels of A-to-I RNA editing across individuals in coding sequences and non-conserved Alu repeats. BMC Genomics 11: 608.

Gu, T., F. W. Buaas, A. K. Simons, C. L. Ackert-Bicknell, R. E. Braun *et al.*, 2012 Canonical A-to-I and C-to-U RNA editing is enriched at 3′UTRs and microRNA target sites in multiple mouse tissues. PLoS One 7: e33720.

Gurevich, I., H. Tamir, V. Arango, A. J. Dwork, J. J. Mann *et al.*, 2002 Altered editing of serotonin 2C receptor pre-mRNA in the prefrontal cortex of depressed suicide victims. Neuron 34: 349–356.

Hassan, M. A., V. Butty, K. D. Jensen, and J. P. Saeij, 2014 The genetic basis for individual differences in mRNA splicing and APOBEC1 editing activity in murine macrophages. Genome Res. 24: 377–389.

Herbert, A., and A. Rich, 2001 The role of binding domains for dsRNA and Z-DNA in the in vivo editing of minimal substrates by ADAR1. Proc. Natl. Acad. Sci. USA 98: 12132–12137.

Hirano, K., S. G. Young, R. V. Farese, Jr., J. Ng, E. Sande *et al.*, 1996 Targeted disruption of the mouse apobec-1 gene abolishes apolipoprotein B mRNA editing and eliminates apolipoprotein B48. J. Biol. Chem. 271: 9887–9890.

Hofacker, I. L., B. Priwitzer, and P. F. Stadler, 2004 Prediction of locally stable RNA secondary structures for genome-wide surveys. Bioinformatics 20: 186–190.

Huang, S., J. Holt, C. Kao, L. McMillan, and W. Wang, 2014 A novel multi-alignment pipeline for high-throughput sequencing data. Database (Oxford), bau057: 1–12.

Keane, T. M., L. Goodstadt, P. Danecek, M. A. White, K. Wong *et al.*, 2011 Mouse genomic variation and its effect on phenotypes and gene regulation. Nature 477: 289–294.

Kiran, A. M., J. J. O'Mahony, K. Sanjeev, and P. V. Baranov, 2013 Darned in 2013: inclusion of model organisms and linking with Wikipedia. Nucleic Acids Res. 41: D258–D261.

Langmead, B., C. Trapnell, M. Pop, and S. L. Salzberg, 2009 Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biol. 10: R25.

Lehmann, K. A., and B. L. Bass, 1999 The importance of internal loops within RNA substrates of ADAR1. J. Mol. Biol. 291: 1–13.

Li, B., and C. N. Dewey, 2011 RSEM: accurate transcript quantification from RNA-seq data with or without a reference genome. BMC Bioinformatics 12: 323.

Li, J. B., E. Y. Levanon, J. K. Yoon, J. Aach, B. Xie *et al.*, 2009 Genome-wide identification of human RNA editing sites by parallel DNA capturing and sequencing. Science 324: 1210–1213.

Lorenz, R., S. H. Bernhart, C. Honer Zu Siederdissen, H. Tafer, C. Flamm *et al.*, 2011 ViennaRNA Package 2.0. Algorithms Mol. Biol. 6: 26.

Munger, S. C., N. Raghupathy, K. Choi, A. K. Simons, D. M. Gatti *et al.*, 2014 RNA-seq alignment to individualized genomes improves transcript abundance estimates in multiparent populations. Genetics 198: 59–73.

Nakamuta, M., K. Oka, J. Krushkal, K. Kobayashi, M. Yamamoto *et al.*, 1995 Alternative mRNA splicing and differential promoter utilization determine tissue-specific expression of the apolipoprotein B mRNA-editing protein (Apobec1) gene in mice: structure and evolution of Apobec1 and related nucleoside/nucleotide deaminases. J. Biol. Chem. 270: 13042–13056.

Ota, H., M. Sakurai, R. Gupta, L. Valente, B. E. Wulff *et al.*, 2013 ADAR1 forms a complex with Dicer to promote microRNA processing and RNA-induced gene silencing. Cell 153: 575–589.

Paz, N., E. Y. Levanon, N. Amariglio, A. B. Heimberger, Z. Ram *et al.*, 2007 Altered adenosine-to-inosine RNA editing in human cancer. Genome Res. 17: 1586–1595.

Petersen-Mahrt, S. K., and M. S. Neuberger, 2003 In vitro deamination of cytosine to uracil in single-stranded DNA by apolipoprotein B editing complex catalytic subunit 1 (APOBEC1). J. Biol. Chem. 278: 19583–19586.

Pickrell, J. K., Y. Gilad, and J. K. Pritchard, 2012 Comment on "Widespread RNA and DNA sequence differences in the human transcriptome." Science 335: 1302 (author reply: 1302).

Ramaswami, G., and J. B. Li, 2014 RADAR: a rigorously annotated database of A-to-I RNA editing. Nucleic Acids Res. 42: D109–D113.

Rieder, L. E., and R. A. Reenan, 2012 The intricate relationship between RNA structure, editing, and splicing. Semin. Cell Dev. Biol. 23: 281–288.

Rosenberg, B. R., C. E. Hamilton, M. M. Mwangi, S. Dewell, and F. N. Papavasiliou, 2011 Transcriptome-wide sequencing reveals numerous APOBEC1 mRNA-editing targets in transcript 3′ UTRs. Nat. Struct. Mol. Biol. 18: 230–236.

Sapiro, A. L., P. Deng, R. Zhang, and J. B. Li, 2015 Cis regulatory effects on A-to-I RNA editing in related Drosophila species. Cell Reports 11: 697–703.

Smith, H. C., R. P. Bennett, A. Kizilyer, W. M. McDougall, and K. M. Prohaska, 2012 Functions and regulation of the APOBEC family of proteins. Semin. Cell Dev. Biol. 23: 258–268.

Svenson, K. L., D. M. Gatti, W. Valdar, C. E. Welsh, R. Cheng *et al.*, 2012 High-resolution genetic mapping using the Mouse Diversity outbred population. Genetics 190: 437–447.

Waterston, R. H., K. Lindblad-Toh, E. Birney, J. Rogers, J. F. Abril *et al.*, 2002 Initial sequencing and comparative analysis of the mouse genome. Nature 420: 520–562.

Yalcin, B., K. Wong, A. Agam, M. Goodson, T. M. Keane *et al.*, 2011 Sequence-based characterization of structural variation in the mouse genome. Nature 477: 326–329.

Yalcin, B., D. J. Adams, J. Flint, and T. M. Keane, 2012 Next-generation sequencing of experimental mouse strains. Mamm. Genome 23: 490–498.

Communicating editor: E. G. Petretto

GENETICS

Supporting Information
www.genetics.org/lookup/suppl/doi:10.1534/genetics.115.179481/-/DC1

Genetic Architectures of Quantitative Variation in RNA Editing Pathways

Tongjun Gu, Daniel M. Gatti, Anuj Srivastava, Elizabeth M. Snyder, Narayanan Raghupathy, Petr Simecek, Karen L. Svenson, Ivan Dotu, Jeffrey H. Chuang, Mark P. Keller, Alan D. Attie, Robert E. Braun, and Gary A. Churchill

DOI: 10.1534/genetics.115.179481

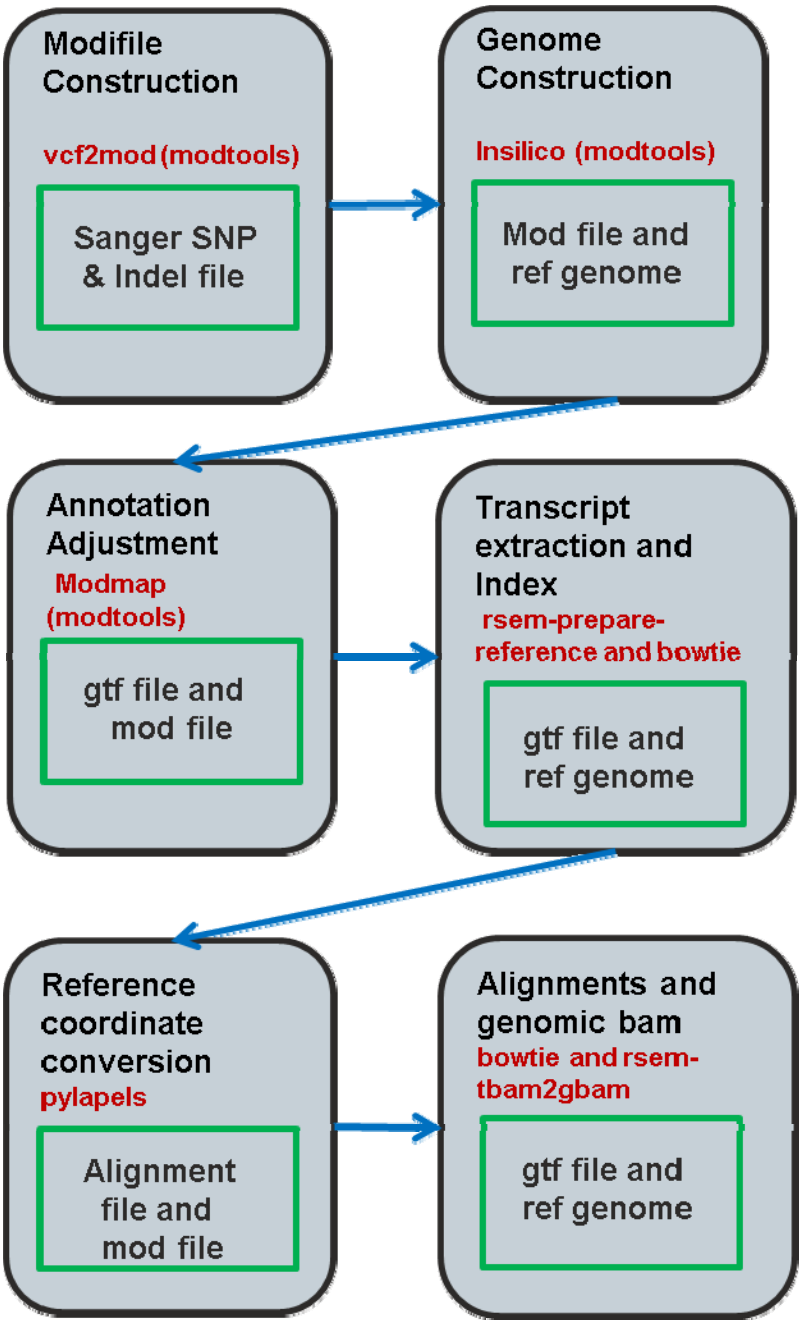

Figure S1. RNASeq alignment pipeline aligns reads from each founder to a founder-specific transcriptome that inserts SNPs and Indels from the Sanger Mouse Genomes Project into the reference sequence.

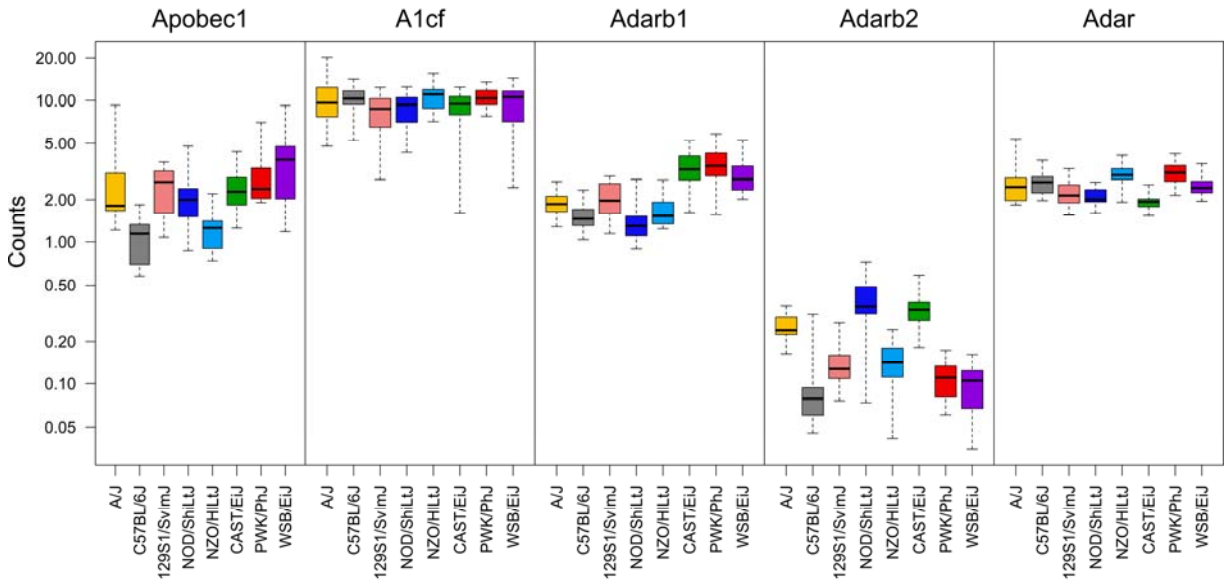

Figure S2. Expression levels in the DO founders for RNA editing related genes. The y-axis shows the upper-quartile normalized total counts for each gene.

Table S1

Canonical and non-canonical editing sites during the denovo discovery process

	A/J	C57BL/6J	129S1/SvImJ	NOD/ShiLtJ	NZO/HlLtJ	CAST/EiJ	PWK/PhJ	WSB/EiJ
All Loci	72042664	74352911	72327688	71984593	72429702	71888532	71921235	71057464
All Hets	5863835	6074674	5983105	5950016	5721370	6122205	5853645	5596353
Hets > 50% repl	11996	11570	11266	13280	10126	10661	9901	11808
MAF > 5%	825	743	716	861	679	1467	1278	821
Hets with 2 alleles	813	732	703	849	667	1449	1263	801
Hets > 160 cov	805	728	700	847	661	1439	1261	784
sites	623	591	582	680	562	1105	966	617
canonical sites	134	136	141	120	149	233	246	168
non-canonical sites	234	193	211	245	192	381	350	223
filter Sanger	94	89	95	106	94	181	206	131
non-canonical sites	119	124	114	137	104	216	217	143

Table S2. Location, genes, founder counts and QTL information for 183 RNA Editing sites. (.xls, 145 KB)

Available for download as a .xlx file at www.genetics.org/lookup/suppl/doi:10.1534/genetics.115.179481/-/DC1/TableS2.xls

Table S3. Mooring sequences of 59 C-to-U editing sites with QTL at the A*pobec1* locus on Chr 6. The motif start is the number of bp from the edited C nucleotide where the mooring sequence starts. The MEME p-value for each site represents the probability that a random sequence with the same length and sequence background would match the discovered motif with a score greater than or equal to the current site. The table is sorted by p-value.

Site	Start	p-value	Pproximal	Mooring	Distal	Ensembl	Symbol
chrX_38420630	0	6.59E-05	.	CTCAGAGATG	GCACCAATTG	ENSMUSG00000016534	Lamp2
chr1_139523522	0	0.000172	.	CTCAATACTT	TCCTTATTTA	ENSMUSG00000026368	F13b
chrX_139672470	12	0.000302	GAATTGGTTG	ATCAGTATAT	TAGTGACAC	ENSMUSG00000031438	Rnf128
chr15_99408620	0	0.000302	.	CTTAGTGGTT	TTCTCTATTT	ENSMUSG00000023010	Tmbim6
chr12_8008054	7	0.000302	CAATTTG	ATCAGTATAT	TAAAGATAAT	ENSMUSG00000020609	Apob
chr3_73638664	7	0.000451	CACTTAG	CTCAATGACT	AATAAATAGG	ENSMUSG00000027792	Bche
chr14_21448058	9	0.000451	CTTAGAGGG	CTCAGTGCTA	CTTCTAGGAC	ENSMUSG00000039197	Adk
chr3_57835070	0	0.000512	.	CTTAGTGACG	TGGAAAATTG	ENSMUSG00000036503	Rnf13
chr10_97517941	0	0.000512	.	CTTAGTAAAG	CGTTAAAAGG	ENSMUSG00000019929	Dcn
chr7_132557671	8	0.000687	CATTTATT	TTCAGTATTT	CTTTGAATAA	ENSMUSG00000030934	Oat
chr3_144597012	0	0.000848	.	CTTAGTTTTG	CATGCTTACA	ENSMUSG00000037072	Sep15
chr9_52088609	7	0.00103	CCTACTA	ATGAGTATTT	GGTAATTTCC	ENSMUSG00000032050	Rdx
chr19_44395692	9	0.00171	CATAGAAAA	ATCACTGTAG	ATCTACTGAC	ENSMUSG00000037071	Scd1
chr14_73362559	17	0.00217	GGATTTCAGC	CTTGATGTTT	TAAT	ENSMUSG00000022108	Itm2b
chrX_60224300	20	0.00246	AGTATTACTT	TTGAGTATTT	T	ENSMUSG00000062949	Atp11c
chr5_87555712	21	0.00246	GGCTTTATAA	CACAGTATAT		ENSMUSG00000029273	Sult1d1
chr4_107200739	0	0.00246	.	CTTAGTTAGT	TGCATTGGTT	ENSMUSG00000028618	Tmem59
chrX_36604963	9	0.00285	CATGGTGAT	TTCAATATTT	AGAAAAGTTT	ENSMUSG00000006373	Pgrmc1
chr16_21761768	10	0.00285	CAAAATGTTG	CTAAGAGAAT	AATTCATAAC	ENSMUSG00000022853	Ehhadh
chr1_58200408	9	0.00337	CATGTTTAA	TTCAATATAT	TAAACGGAAT	ENSMUSG00000064294	Aox3
chr10_97517901	7	0.00386	CTGAGTT	ATCAAAGTCT	GATGTAATCA	ENSMUSG00000019929	Dcn
chr9_72747608	1	0.00545	C	TTTAATGATT	AGTATACTGC	ENSMUSG00000032216	Nedd4
chr7_131445518	15	0.00545	TAGAAAGTAT	TTTGGTGCTT	TTGCAA	ENSMUSG00000030861	Acadsb
chr7_131445515	18	0.00545	TAGAAAGTAT	TTTGGTGCTT	TTG	ENSMUSG00000030861	Acadsb

chr10_57515985	19	0.00545	TAGTATGCCA	CTTAATTAAG	AT	ENSMUSG00000019877	Serinc1
chr8_45306577	0	0.006	.	CTTGGAAATT	CATATATCAT	ENSMUSG00000079057	Cyp4v3
chr11_20063380	13	0.00669	AATGATGATT	CTGAGAGAAA	TATTTTCC	ENSMUSG00000020152	Actr2
chr9_79769822	8	0.00752	CAGTTTAG	ATTAATATGT	GCTTAAAAGA	ENSMUSG00000032328	Tmem30a
chr9_114749171	17	0.00752	AATTACTATA	CTTAAATTAT	GCTC	ENSMUSG00000032434	Cmtm6
chr5_17783137	6	0.00752	CAAAAG	ATTAATATGT	CACTATAGGC	ENSMUSG00000002944	Cd36
chr3_121760011	0	0.00752	.	CTAAGAATGT	CCTTATTCTT	ENSMUSG00000028127	Abcd3
chr1_139810468	19	0.00752	AATATGATTT	CACAAAGCAT	GT	ENSMUSG00000033898	Cfhr2
chrX_12616481	10	0.00836	CTATGTTGAT	TTCAGAAGAG	ACTAGCTTTG	ENSMUSG00000031007	Atp6ap2
chr7_131444739	0	0.0103	.	CTTAGTTCTA	AATATCAAAA	ENSMUSG00000030861	Acadsb
chr1_139810666	18	0.0103	AAAATGCAAA	AGCAGTAATT	CAA	ENSMUSG00000033898	Cfhr2
chr10_97518033	4	0.0103	CTCT	AGCAATGTAT	TAATCTCCTT	ENSMUSG00000019929	Dcn
chr10_7768196	7	0.0103	CTATTTG	ATTACTATTG	TAAGATTTTA	ENSMUSG00000040006	BC013529
chr9_100982512	4	0.0113	CTTT	TTGAATAAAT	ATTTCATTGT	ENSMUSG00000032527	Pccb
chr8_95864156	7	0.0113	CAATAAA	ATTACTATAG	ACCCAG	ENSMUSG00000031672	Got2
chr5_145854741	12	0.0123	AAAATTTTTG	ATGAAAAATG	TGAGCTCTT	ENSMUSG00000056035	Cyp3a11
chrX_38419909	2	0.0146	CA	TTTAGATTAT	ATATCGGATT	ENSMUSG00000016534	Lamp2
chr2_122152902	11	0.0146	TTCTCATTAC	TTGGATGCAG	TTACTCATCT	ENSMUSG00000060802	B2m
chr14_73362604	8	0.0146	CTACTTTA	ATTAAAATGT	GCCGTATCTT	ENSMUSG00000022108	Itm2b
chr14_21448185	4	0.0159	CATT	TTCAATTGTT	TGTAAATTCA	ENSMUSG00000039197	Adk
chr13_95627449	16	0.0159	CACTTTCTGT	ATAAAAGTAT	ATATT	ENSMUSG00000021676	Iqgap2
chr13_93752703	2	0.0174	CA	TTAGGTGTTT	GCCAATATGA	ENSMUSG00000042102	Dmgdh
chr8_13174652	19	0.0188	AATCTGCTTT	ATCAAATGTG	AA	ENSMUSG00000031447	Lamp1
chr4_59618515	6	0.0203	CAATGC	CTGGCTTTAT	TGAGCTTTCA	ENSMUSG00000028383	Hsdl2
chr18_60391275	0	0.0203	.	CTTAGATCAA	GTAATTTTAC	ENSMUSG00000054072	Iigp1
chr7_14411655	11	0.0252	TTGTTTTTGA	TTTAAGGATG	TGGCATATAA	ENSMUSG00000030378	2810007J24Rik
chr8_13174720	8	0.0271	CTATTGAA	ATGACGGTGT	TAATTTTGCT	ENSMUSG00000031447	Lamp1
chr3_119432749	3	0.0271	CAC	ATAAATTCAT	TTATTCCTTC	ENSMUSG00000033308	Dpyd
chr6_138156528	7	0.031	CAGTAGG	CTCTATTCTT	TTGTATTTGG	ENSMUSG00000008540	Mgst1
chr6_87999680	7	0.0331	CAGTAGA	CACAAGAATT	ATGTACGCCT	ENSMUSG00000079477	Rab7

chr6_52546355	3	0.0331	CTC	CAGAAAAATG	ACACCTTTAG	ENSMUSG00000029776	Hibadh
chr6_141908813	4	0.0353	CCTC	AACAATTATT	TTTTACTCAT	ENSMUSG00000041698	Slco1a1
chrX_109161347	6	0.0424	CTTCAA	ATTACTATTA	TCATCATACC	ENSMUSG00000031246	Sh3bgrl
chr16_56016950	6	0.0621	CACATG	ATTAGTTTCC	AAGGGTTACA	ENSMUSG00000071533	Pcnp
chrX_53021282	13	0.0653	TTGATTTGCA	CTATGAGCCT	ATAGGCCA	ENSMUSG00000025630	Hprt

File S1

RNA Seq Alignment Pipeline

Building transcriptome of CC founders (except B6) using modtools.

Data needed:

SNPs and Indels:

Site: ftp://ftp-mouse.sanger.ac.uk/REL-1303-SNPs_Indels-GRCm38/

Files Downloaded: mgp.v3.snps.rsIDdbSNPv137.vcf.gz, mgp.v3.indels.rsIDdbSNPv137.vcf.gz

Note: Download the *.tbi files too

Reference genome:

ftp://ftp-mouse.sanger.ac.uk/ref/

File downloaded: GRCm38_68.fa

Header modified:

GRCm38_68_mod.fa (Only 1, 2, 3, 4…… present in chromosome name)

GRCm38_68_chr.fa (chr present in chromosome name)

Annotation file:

File downloaded: ftp://ftp.ensembl.org/pub/release-68/gtf/mus_musculus/Mus_musculus.GRCm38.68.gtf.gz

gunzip (Mus_musculus.GRCm38.68.gtf.gz)

Modified file: Mus_musculus.GRCm38.68_final.gtf

(header "##" is removed and "chr" is added to chromosome name ; chrMT is replaced by chrM)

In order to modtool to work download following two files:

Site: http://csbio.unc.edu/~sphuang/vcf2mod/

Files downloaded: sanger_vcf.alias, build38.meta

Important Information:

1. All Strain Name in Sanger VCF and Indel build38 files:

129P2	129S1	129S5	AJ	AKRJ	BALBcJ	C3HHeJ	C57BL6NJ	CASTEiJ
CBAJ	DBA2J	FVBNJ	LPJ	NODShiLtJ	NZOHlLtJ	PWKPhJ	SPRETEiJ	WSBEiJ

Step1: (Convert vcf files to mod file)

Module requirement:

- samtools
- tabix
- vcf-tools
- python (with modtools installed)

Command:

vcf2mod -a sanger_vcf.alias build38 build38.meta \

129S1 mgp.v3.snps.rsIDdbSNPv137.vcf.gz \

mgp.v3.indels.rsIDdbSNPv137.vcf.gz

Input:

- sanger_vcf.alias (vcf alias file with 1 for chr1, 2 for chr2 … so on; downloaded for link mentioned above)

- build38 (just type this phrase ; needed for modtools)
- build38.meta (build38 meta-data file; downloaded for link mentioned above)
- CC founder strain name mentioned exactly as in Sanger vcf file; 129S1 here [See Table 1]
- Sanger SNP file
- Sanger Indel file

Output:

- 129S1.mod
- 129S1.mod.tbi

Step2: Using the Mod file to create the pseudogenome

Module requirement:

- samtools
- tabix
- vcf-tools
- python (with modtools installed)

Command:

insilico 129S1.mod \

GRCm38_68_mod.fa \

-a sanger_vcf.alias –o 129S1_pseudo.fa

Input:

- 129S1.mod (vcf2mod file)
- GRCm38_68_mod.fa (GRCm38_68.fa header modified file)
- sanger_vcf.alias (alias file)

Output:

- 129S1_pseudo.fa

Step3: Adjusting the annotations using modmap

Module requirement:

- samtools
- tabix
- vcf-tools
- python (with modtools installed)

Command:

1. **Left Coordinate Fix:**

modmap -d '\t' 129S1.mod Mus_musculus_final.GRCm38.68.gtf 129S1_Mus_musculus_final_left_fix.GRCm38.68.gtf 1,4

2. **Right Coordinate Fix:**

modmap -d '\t' 129S1.mod \ 129S1_Mus_musculus_final_left_fix.GRCm38.68.gtf \129S1_Mus_musculus_final_left_right_fix.GRCm38.68.gtf 1,5

3. **Taking absolute value:**

Unix commands: 129S1_Mus_musculus_final_left_right_fix.GRCm38.68.gtf \

>129S1_Mus_musculus_final_left_right_abs_fix.GRCm38.68.gtf

Note: Take absolute value for negative positions [some negative position values from modmap output. This is because the requested position has no corresponding position in the pseudogenome (due to deletion). In this case, as suggested by author of modtools, we can take the absolute value of that negative position value, and this will give us the nearest corresponding position in the psedogenome]

Input:

Mus_musculus_final.GRCm38.68.gtf (location given above)

Output:

Mus_musculus_final_left_right_abs_fix.GRCm38.68.gtf

Step4: extract transcriptome and build bowtie index

Module requirement:

samtools

rsem/1.2.15

bowtie/0.12.8

Commands:

```
rsem-prepare-reference  \
--gtf 129S1_Mus_musculus_final_left_right_abs_fix.GRCm38.68.gtf \
--no-polyA  \
--bowtie  129S1_pseudo.fa  129S1_pseudo
```

Input:

gtf file: 129S1_Mus_musculus_final_left_right_abs_fix.GRCm38.68.gtf

pseudogenome file: 129S1_pseudo.fa

basename for bowtie index: 129S1_pseudo

Output:

- 129S1_pseudo.1.ebwt
- 129S1_pseudo.2.ebwt
- 129S1_pseudo.3.ebwt
- 129S1_pseudo.4.ebwt
- 129S1_pseudo.chrlist
- 129S1_pseudo.fa
- 129S1_pseudo.grp
- 129S1_pseudo.idx.fa
- 129S1_pseudo.n2g.idx.fa
- 129S1_pseudo.rev.1.ebwt

- 129S1_pseudo.rev.2.ebwt
- 129S1_pseudo.seq
- 129S1_pseudo.ti
- 129S1_pseudo.transcripts.fa

Note: Step1 to Step4 Needed for each founder strain except B6

B6 Transcriptome building:

- samtools
- tabix
- rsem/1.2.15
- bowtie/0.12.8

Commands:

rsem-prepare-reference --gtf Mus_musculus.GRCm38.68_final.gtf --no-polyA --bowtie GRCm38_68_chr.fa GRCm38_68_chr

Input:

gtf file: Mus_musculus.GRCm38.68_final.gtf

pseudogenome file: GRCm38_68_chr.fa

basename for bowtie index: GRCm38_68_chr

Output:

- GRCm38_68_chr.1.ebwt
- GRCm38_68_chr.2.ebwt
- GRCm38_68_chr.3.ebwt
- GRCm38_68_chr.4.ebwt
- GRCm38_68_chr.chrlist
- GRCm38_68_chr.fa
- GRCm38_68_chr.fa.fai
- GRCm38_68_chr.grp

- GRCm38_68_chr.idx.fa
- GRCm38_68_chr.n2g.idx.fa
- GRCm38_68_chr.rev.1.ebwt
- GRCm38_68_chr.rev.2.ebwt
- GRCm38_68_chr.seq
- GRCm38_68_chr.ti
- GRCm38_68_chr.transcripts.fa

File S2

Pileup Ensembl Genes in DO Founders

```
#########################################################################
##########
# Pileup the BAM files for each founder separately. Save locations
with > 1
# read and at least one heterozygous read in one founder.
# This is intended to cast a wide net and retain locations that can be
filtered
# in downstream steps.
# PileupParams: minBaseQual = 20
#               maxDepth = 1e7
#
# Daniel Gatti
# dan.gatti@jax.org
# Nov. 5, 2014
# Ensembl 68
# Sanger v3
#########################################################################
##########
library(GenomicRanges)
library(Rsamtools)

strain = commandArgs(trailingOnly = T)

setwd("/hpcdata/dgatti/RNAediting/")

# Load in the ensembl 68 GTF.
load("/hpcdata/cgd/ensembl/release68/Mus_musculus.GRCm38.68.Rdata")

# Add 'chr' to the chromosome names to match the BAM files.
ensembl = keepSeqlevels(x = ensembl, value =
          seqlevels(ensembl)[-grep("JH|GL", seqlevels(ensembl))])
ensembl = renameSeqlevels(x = ensembl, value = paste0("chr",
seqlevels(ensembl)))
sl = sub("MT", "M", seqlevels(ensembl))
ensembl = renameSeqlevels(x = ensembl, value = sl)

# Keep only unique exons and UTRs.
length(unique(ensembl$gene_id))
ensembl = ensembl[ensembl$feature == "exon"]
keys = paste0(seqnames(ensembl), start(ensembl), end(ensembl))
ensembl = ensembl[!duplicated(keys)]
length(unique(ensembl$gene_id))

# Get the BAM files in Anuj's directory.
bamdir =
"/hpcdata/anuj/Projects/Investigator/Gary_ChurChill/RNA_editing/Final_
DataSet/FounderBams"
```

```
bamfiles = dir(path = bamdir, pattern = "bam$", full.names = T)
bamfiles = bamfiles[grep(strain, bamfiles)]

# Make a list of PileupFiles by strain.
pufiles = PileupFiles(bamfiles)

# Make the pileup function.
pufxn = function(x) {
          dimnames(x$seq)[[3]] = x$pos
          x$seq
        } # pufxn()

param = ApplyPileupsParam(minBaseQual = 20L, maxDepth = 1e7,
        what = "seq", which = ensembl)

setwd("founders")

print(paste(strain, date()))

# Pileup all 16 replicates.
pileup = applyPileups(files = pufiles, FUN = pufxn, param = param)
names(pileup) = paste(ensembl$gene_id, ensembl$transcript_id,
                      ensembl$exon_number, sep = "_")

save(pileup, file = paste0("_1_", strain, "_pileup.Rdata"))
```

File S3

Filter Sites by Coverage and Minor Allele Frequency

```
##########################################################################
##########
# Read in the pileups for each strain and filter by:
# minimum coverage per site in each sample > 1% of total reads
# alternate allele present in > 50% of samples
# total coverage across all 16 samples >= 160
#
#
# Daniel Gatti
# dan.gatti@jax.org
# Nov. 5, 2014
# Ensembl 68
# Sanger v3
##########################################################################
##########
library(GenomicRanges)
library(Rsamtools)
library(BSgenome.Mmusculus.UCSC.mm10)

strain = commandArgs(trailingOnly = T)

print(paste(strain, date()))

setwd("/hpcdata/dgatti/RNAediting/")

# Load in the ensembl 68 GTF.
load("/hpcdata/cgd/ensembl/release68/Mus_musculus.GRCm38.68.Rdata")

# Add 'chr' to the chromosome names to match the BAM files.
ensembl = keepSeqlevels(x = ensembl, value =
          seqlevels(ensembl)[-grep("JH|GL", seqlevels(ensembl))])
ensembl = renameSeqlevels(x = ensembl, value = paste0("chr",
seqlevels(ensembl)))
sl = sub("MT", "M", seqlevels(ensembl))
ensembl = renameSeqlevels(x = ensembl, value = sl)

# Load in the pileup file for this strain.
setwd("founders")
# This loads in a list called 'pileup'.
# Each element is a 3D array: num_alleles x num_samples x
num_positions.
load(file = paste0("_1_", strain, "_pileup.Rdata"))

################################################
# Adjust these parameters to change the filters.
varMAF = 0.05
minCov = 2
```

```
replicateAlleleFreq = 0.75
totalCov = 160

# Keep only loci with sum > 0.
pileup = pileup[sapply(pileup, sum) > 0]

print(paste("ALL LOCI:", sum(sapply(pileup, dim)[3,])))

# Keep only positions with more than one allele call.
pileup = lapply(pileup, function(z) {
            keep = matrix(F, dim(z)[1], dim(z)[3])
            # For each position, sum the number of expressed alleles.
            for(j in 1:dim(z)[3]) { keep[,j] = rowSums(z[,,j]) > 0 }
            z[,,colSums(keep) > 1, drop = F]
          })

pileup = pileup[sapply(pileup, sum) > 0]
print(paste("ALL HETS:", sum(sapply(pileup, dim)[3,])))

# Set alleles which do not have minCov reads in at least 75% of
replicates = 0.
length(pileup)
pileup = lapply(pileup, function(z) {
            # Set cells with cov <= 2 in < 75% of replicates = 0.
            keep = apply(z > minCov, 3, rowMeans) > replicateAlleleFreq
            z = sweep(z, c(1,3), keep, "*")
            # Remove positions with 0 reads.
            for(j in 1:dim(z)[3]) { keep[,j] = rowSums(z[,,j]) > 0 }
            z[,,colSums(keep) > 1, drop = F]
          })

pileup = pileup[sapply(pileup, sum) > 0]
print(paste("HETS in > 50% repl:", sum(sapply(pileup, dim)[3,])))

# Sum reads across all replicates. This produces matrices with
# dimensions num_alleles * num_positions.
pileup = lapply(pileup, apply, c(1,3), sum)

# Set cells with variant minor allele frequency <= varMAF = 0.
length(pileup)
pileup = lapply(pileup, function(z) {
            # Zero out cells with MAF < 5%.
            z = t(z)
            keep = z / rowSums(z) > varMAF
            z = z * keep
            # Keep only het positions.
            t(z[rowSums(z > 0) > 1,,drop = F])
          })

pileup = pileup[sapply(pileup, sum) > 0]
print(paste("HETS w/ MAF > 5%:", sum(sapply(pileup, ncol))))
```

```
# Remove sites that have more than 2 alleles.
pileup = lapply(pileup, function(z) {
           z[,colSums(z > 0) == 2, drop = F]
         })

pileup = pileup[sapply(pileup, sum) > 0]
print(paste("HETS w/ 2 alleles:", sum(sapply(pileup, ncol))))

# Filter by total coverage at each site.
length(pileup)
pileup = lapply(pileup, function(z) {
           z[,colSums(z) >= totalCov, drop = F]
         })
pileup = pileup[sapply(pileup, sum) > 0]

print(paste("HETS >= totalCov:", sum(sapply(pileup, ncol))))

# Condense the data into a table with chr, position, gene ID and
# counts.
len = sapply(pileup, ncol)
df = data.frame(chr = rep(NA, sum(len)),
                pos = rep(0, sum(len)),
                siteid = rep(NA, sum(len)),
                gene_id = rep(NA, sum(len)),
                symbol = rep(NA, sum(len)),
                strand = rep(NA, sum(len)),
                ref = rep(NA, sum(len)),
                A = rep(0, sum(len)),
                C = rep(0, sum(len)),
                G = rep(0, sum(len)),
                T = rep(0, sum(len)))

len = c(0, cumsum(len))
names = strsplit(names(pileup), split = "_")
for(i in 1:length(pileup)) {

  rng = (len[i] + 1):len[i+1]
  df$gene_id[rng] = rep(names[[i]][1], length(rng))

  ens = ensembl[ensembl$gene_id == names[[i]][1]]
  df$chr[rng] = rep(as.character(runValue(seqnames(ens))[1]),
length(rng))
  df$pos[rng] = as.numeric(colnames(pileup[[i]]))
  df$symbol[rng] = rep(ens$gene_name[1], length(rng))
  df$strand[rng] = rep(as.character(runValue(strand(ens))[1]),
length(rng))
  df[rng,(ncol(df) - 3):ncol(df)] = t(pileup[[i]][-5,])

} # for(i)

# Add the reference base.
grtmp = GRanges(seqnames = df$chr,
```

```
        ranges = IRanges(start = df$pos, width = 1), strand =
df$strand)
df$ref = as.character(getSeq(BSgenome.Mmusculus.UCSC.mm10, grtmp))
rm(grtmp)

# Change the counts for genes on the '-' strand to their complementary
bases.
minus = which(df$strand == "-")
df[minus,(ncol(df) - 3):ncol(df)] = df[minus, ncol(df):(ncol(df) - 3)]

# Set the site ID.
df$siteid = paste(df$chr, df$pos, sep = "_")

# Keep only unique sites.
sites = df[!duplicated(df$siteid),]
rm(df)

# Remove sites where the reference base does not have any reads.
# Is this due to differenes in the source of the reference?
keep = rep(T, nrow(sites))
for(i in 1:nrow(sites)) {
  keep[i] = sites[i,sites$ref[i]] > 0
} # for(i)
sites = sites[keep,]

print(paste("Unique sites:", nrow(sites)))

save(sites, file = paste0("_2_", strain, "_het_sites.Rdata"))

print(paste(strain, date()))
```

File S4

Filter out non-canonical editing sites

```
#########################################################################
# Filter the sites for each strain by removing all sites for any gene
# that has a non-canonical editing site.
#
# Daniel Gatti
# Dan.Gatti@jax.org
# Nov. 14, 2014
# Ensembl 68
# Sanger v3
#########################################################################
##########
library(GenomicRanges)
library(Rsamtools)
library(BSgenome.Mmusculus.UCSC.mm10)

strain = commandArgs(trailingOnly = T)

print(paste(strain, date()))

setwd("/hpcdata/dgatti/RNAediting/founders/")

# Load in the ensembl 68 GTF.
load("/hpcdata/cgd/ensembl/release68/Mus_musculus.GRCm38.68.Rdata")

# Add 'chr' to the chromosome names to match the BAM files.
ensembl = keepSeqlevels(x = ensembl, value =
          seqlevels(ensembl)[-grep("JH|GL", seqlevels(ensembl))])
ensembl = renameSeqlevels(x = ensembl, value = paste0("chr",
seqlevels(ensembl)))
ensembl = renameSeqlevels(x = ensembl, value = sub("MT", "M",
seqlevels(ensembl)))

# Load in the het sites for this strain. This loads in a data.frame
# called 'sites'.
load(file = paste0("_2_", strain, "_het_sites.Rdata"))

edit.type = rep("", nrow(sites))
for(i in 1:nrow(sites)) {

    wh = which(sites[i,8:11] > 0)
    edit.type[i] = paste(sort(colnames(sites)[8:11][wh]), collapse =
"")

} # for(i)

sites = data.frame(sites, edit.type)
```

```
# Split up the sites by gene.
sites = split(sites, sites$gene_id)

# Keep the sites that only have cannonical editing.
keep = rep(F, length(sites))
for(i in 1:length(sites)) {
  keep[i] = all(sites[[i]]$edit.type %in% c("AG", "CT"))
} # for(i)

# Keep non-canonical sites.
noncanon.sites = sites[!keep]

# Keep canonical sites.
sites = sites[keep]

# Join the sites back together.
sites = unsplit(sites, rep(names(sites), sapply(sites, nrow)))
noncanon.sites = unsplit(noncanon.sites, rep(names(noncanon.sites),
                 sapply(noncanon.sites, nrow)))
noncanon.sites = noncanon.sites[!noncanon.sites$edit.type %in% c("AG",
"CT"),]

print(paste("Canonical edit sites:", nrow(sites)))
print(paste("Non-canonical edit sites:", nrow(noncanon.sites)))
print(paste("Unique genes:", length(unique(sites$gene_id))))

save(sites, file = paste0("_3_", strain,
"_canonical_edit_sites.Rdata"))
save(noncanon.sites, file = paste0("_3_", strain,
"_non_canonical_edit_sites.Rdata"))

print(paste(strain, date()))
```

File S5

Remove sites that intersect with genomic variants in the founders

```
##########################################################################
##########
# Intersect the heterozygous positions for each strain with the Sanger
SNPs
# Indels and SVs.
#
# Daniel Gatti
# dan.gatti@jax.org
# Nov. 5, 2014
# Ensembl 68
# Sanger v3
##########################################################################
##########
library(GenomicRanges)
library(Rsamtools)

strain = commandArgs(trailingOnly = T)

print(paste(strain, date()))

# Set the paths to the Sanger variant files.
snpfile = "/hpcdata/cgd/Sanger/REL-1410/mgp.v4.snps.dbSNP.vcf.gz"
indelfile = "/hpcdata/cgd/Sanger/REL-1410/mgp.v4.indels.dbSNP.vcf.gz"
svfile = "/hpcdata/cgd/Sanger/REL-1302/18strains.REL-1302-SV-
GRCm38.sdp.tab.gz"

setwd("/hpcdata/dgatti/RNAediting/founders/")

# Load in the filtered het sites.
# This loads in a data.frame called 'sites'.
load(file = paste0("_3_", strain, "_canonical_edit_sites.Rdata"))

# The Sanger variant files do not contain the mitochondrial
chromosome.
# Remove it from our data.
rng = which(sites$chr != "chrM")

# Turn the sites into a GRanges object.
gr = GRanges(seqnames = sub("^chr", "", sites$chr[rng]),
             ranges = IRanges(start = sites$pos[rng], width = 1),
             strand = sites$strand[rng], mcols =
sites[rng,c(3:5,7:11)])
colnames(mcols(gr)) = sub("^mcols\\.", "", colnames(mcols(gr)))

######
```

```
# SNPs
tf = TabixFile(snpfile)

# Get the column names.
hdr = headerTabix(tf)
cn = strsplit(hdr$header[length(hdr$header)], split = "\t")[[1]]
cn = sub("^#", "", cn)

# Get the Sanger SNPs.
snps = scanTabix(file = tf, param = gr)
names(snps) = gr$siteid

# Keep only the ones with length > 0.
length(snps)
snps = snps[sapply(snps, length) > 0]
length(snps)

# Parse the SNPs into a data.frame.
snps = lapply(snps, strsplit, split = "\t")
snps = lapply(snps, "[", 1)
snps = matrix(unlist(snps), ncol = length(cn), byrow = T,
       dimnames = list(names(snps), cn))

# Keep only the header and DO founder columns.
snps = snps[,c(1:7, 14, 11, 30, 32, 22, 34, 37)]

# Keep the allele call strings for each founder.
snps[,(ncol(snps)-6):ncol(snps)] = substring(snps[,(ncol(snps)-
6):ncol(snps)], 1, 3)

# Keep only the polymorphic snps. (NOTE: 0/0 is reference call)
dim(snps)
snps = snps[rowSums(snps[,(ncol(snps)-6):ncol(snps)] == "0/0") < 7,]
dim(snps)

# Remove sites that intersect with these SNPs.
removed = sites[sites$siteid %in% rownames(snps),]
dim(sites)
sites = sites[!sites$siteid %in% rownames(snps),]
dim(sites)


########
# Indels
tf = TabixFile(indelfile)

# Get the column names.
hdr = headerTabix(tf)
cn = strsplit(hdr$header[length(hdr$header)], split = "\t")[[1]]
cn = sub("^#", "", cn)

# Get the Sanger Indels.
```

```
indels = scanTabix(file = tf, param = gr)
names(indels) = gr$siteid

# Keep only the ones with length > 0.
length(indels)
indels = indels[sapply(indels, length) > 0]
length(indels)

# Parse the Indels into a data.frame.
indels = lapply(indels, strsplit, split = "\t")
indels = lapply(indels, "[", 1)
indels = matrix(unlist(indels), ncol = length(cn), byrow = T,
       dimnames = list(names(indels), cn))

# Keep only the header and DO founder columns.
indels = indels[,c(1:7, 14, 11, 30, 32, 22, 34, 37)]

# Keep the allele call strings for each founder.
indels[,(ncol(indels)-6):ncol(indels)] =
       substring(indels[,(ncol(indels)-6):ncol(indels)], 1, 3)

# Keep only the polymorphic Indels. (NOTE: ./. is reference call)
dim(indels)
indels = indels[rowSums(indels[,(ncol(indels)-6):ncol(indels)] ==
"./.") < 7,]
dim(indels)

# Remove sites that intersect with these indels.
removed = sites[sites$siteid  %in% rownames(indels),]
dim(sites)
sites   = sites[!sites$siteid %in% rownames(indels),]
dim(sites)

#####
# SVs
tf = TabixFile(svfile)

# Get the column names.
hdr = headerTabix(tf)
cn = strsplit(hdr$header[length(hdr$header)], split = "\t")[[1]]
cn = sub("^#", "", cn)

######
# The Sanger SV file does not contain the Y chromosome.
# Remove it.
gr = gr[seqnames(gr) != "Y",]

# Get the Sanger SVs.
svs = scanTabix(file = tf, param = gr)
names(svs) = gr$siteid

# Keep only the ones with length > 0.
```

```
length(svs)
svs = svs[sapply(svs, length) > 0]
length(svs)

# Parse the SVs into a data.frame.
svs = lapply(svs, strsplit, split = "\t")
svs = lapply(svs, "[", 1)
svs = matrix(unlist(svs), ncol = length(cn), byrow = T,
       dimnames = list(names(svs), cn))

# Keep only the header and DO founder columns.
svs = svs[,c(1:4, 8, 6, 18, 19, 13, 20, 22)]

# Keep only the polymorphic SVs. (NOTE: 0 is reference call)
dim(svs)
svs = svs[rowSums(svs[,(ncol(svs)-6):ncol(svs)] == "0") < 7,]
dim(svs)

# Remove sites that intersect with these svs.
removed = sites[sites$siteid  %in% rownames(svs),]
dim(sites)
sites   = sites[!sites$siteid %in% rownames(svs),]
dim(sites)

print(paste(nrow(sites), "sites in", length(unique(sites$gene_id)),
"genes."))

save(sites, file = paste0("_4_", strain,
"_het_sites_sanger_removed.Rdata"))
```

File S6

Gather common denovo editing sites in the DO founders

```
#########################################################################
##########
# Read in the filtered pileups for each strain and get the union of
all
# edit sites.
#
# Daniel Gatti
# dan.gatti@jax.org
# Nov. 5, 2014
# Ensembl 68
# Sanger v3
#########################################################################
##########
library(GenomicRanges)
library(Rsamtools)
library(BSgenome.Mmusculus.UCSC.mm10)

setwd("/hpcdata/dgatti/RNAediting/")

# Load in the ensembl 68 GTF.
load("/hpcdata/cgd/ensembl/release68/Mus_musculus.GRCm38.68.Rdata")

# Keep the autosomes, X, Y and M.
ensembl = keepSeqlevels(x = ensembl, value =
          seqlevels(ensembl)[-grep("JH|GL", seqlevels(ensembl))])

# Keep only unique exons and UTRs.
length(unique(ensembl$gene_id))
ensembl = ensembl[ensembl$feature == "exon"]
keys = paste0(seqnames(ensembl), start(ensembl), end(ensembl))
ensembl = ensembl[!duplicated(keys)]
length(unique(ensembl$gene_id))

# Load in the filtered pileups for the DO founders.
setwd("founders")

files = dir(pattern = "^_4_|het_sites_sanger_removed.Rdata$")
tmp = vector("list", length(files))
names(tmp) = sub("_4_het_sites_sanger_removed.Rdata$", "", files)
for(i in 1:length(files)) {
  # This loads in an object called 'sites'.
  load(file = files[i])
  tmp[[i]] = sites
} # for(i)
sites = tmp
rm(tmp)
```

```
# Create GRanges objects from the locations and take thier union.
gr = GRanges(seqnames = sites[[1]]$chr,
             ranges = IRanges(start = sites[[1]]$pos, width = 1),
             strand = sites[[1]]$strand)
for(i in 2:length(sites)) {

  gr2 = GRanges(seqnames = sites[[i]]$chr,
                ranges = IRanges(start = sites[[i]]$pos, width = 1),
                strand = sites[[i]]$strand)

  gr = union(gr, gr2)

} # for(i)

# The union() method condenses adjacent sites into a single range.
Remake the
# GRanges object with each site distinct.
pos = unlist(sapply(1:length(gr), function(z) {
start(gr)[z]:end(gr)[z] }))
gr = GRanges(seqnames = rep(as.character(seqnames(gr)), width(gr)),
              ranges = IRanges(start = pos, width = 1),
              strand = rep(as.character(strand(gr)), width(gr)))
gr = renameSeqlevels(gr, sub("chr", "", seqlevels(gr)))
gr = renameSeqlevels(gr, sub("M", "MT", seqlevels(gr)))

print(paste("Union of editing sites:", length(gr)))

# Get the ensembl genes associated with each editing site.
ol = findOverlaps(query = gr, subject = ensembl)
ol = ol[!duplicated(queryHits(ol))]

gr$gene_id = ensembl$gene_id[subjectHits(ol)]
gr$symbol  = ensembl$gene_name[subjectHits(ol)]
gr$ref = rep(NA, length(gr))
gr$edit.type = rep(NA, length(gr))

# We have to change the seqlevels for the mitochondrial chromosome to
# get the reference alleles from the BSGenome.
grtmp = gr
grtmp = renameSeqlevels(grtmp, paste0("chr", seqlevels(grtmp)))
grtmp = renameSeqlevels(grtmp, sub("MT", "M", seqlevels(grtmp)))
seq = as.character(getSeq(BSgenome.Mmusculus.UCSC.mm10, grtmp))
gr$ref = seq
rm(grtmp)

stopifnot(all(as.character(strand(gr)) ==
as.character(strand(ensembl)[subjectHits(ol)])))
stopifnot(all(width(gr) == 1))


###
# Go back into the BAM files and pile up these editing sites.
```

```
# Get the BAM files in Anuj's directory.
bamdir =
"/hpcdata/anuj/Projects/Investigator/Gary_ChurChill/RNA_editing/Final_
DataSet/FounderBams"
bamfiles = dir(path = bamdir, pattern = "bam$", full.names = T)

# Make a list of PileupFiles by strain.
pufiles = PileupFiles(bamfiles)

# Make the pileup function.
pufxn = function(x) {
          dimnames(x$seq)[[3]] = x$pos
          x$seq
        } # pufxn()

# Add "chr" to the seqlevels of the sites.
gr = renameSeqlevels(gr, paste0("chr", sub("MT", "M", seqlevels(gr))))
param = ApplyPileupsParam(minBaseQual = 20L, maxDepth = 1e7,
        what = "seq", which = gr)

# Pileup all 128 samples.
pileup = applyPileups(files = pufiles, FUN = pufxn, param = param)
names(pileup) = paste(seqnames(gr), start(gr), sep = "_")

stopifnot(all(sapply(pileup, dim)[3,] == 1))

save(pileup, file = "all_founder_pileup.Rdata")

# Get the strain names for the pileups.
samples =
gsub("^/hpcdata/anuj/Projects/Investigator/Gary_ChurChill/RNA_editing/
Final_DataSet/FounderBams/|\\.(lapel|unique.sorted).bam$",
          "", colnames(pileup[[1]]))
strains = factor(sapply(strsplit(samples, split = "_"), "[", 1))

# Create a large 3D matrix from the pileup values.
mat = array(unlist(pileup), c(nrow(pileup[[1]]), ncol(pileup[[1]]),
length(pileup)),
      dimnames = list(rownames(pileup[[1]]), colnames(pileup[[1]]),
names(pileup)))

# Complementary bases for genes on the "-" strand.
compl = c("A", "C", "G", "T")
names(compl) = c("T", "G", "C", "A")

# Condense the pileups at each site down to 8 values, one for each
founder.
founders = matrix(0, nrow = dim(mat)[3], ncol =
length(levels(strains)) * 4,
           dimnames = list(dimnames(mat)[[3]],
```

```
           paste(rep(levels(strains), each = 4), c("A","C","G","T"),
sep = "_")))
for(i in 1:dim(mat)[3]) {

  agg = aggregate(t(mat[-5,,i]), list(strains), sum)
  rownames(agg) = agg[,1]
  agg = as.matrix(t(agg[,-1]))

  if(as.character(strand(gr)[i]) == "-") {

    rownames(agg) = compl[rownames(agg)]
    agg = agg[order(rownames(agg)),]

  } # if(as.character(strand(ens)[1]) == "-")

  founders[i,] = agg

} # for(i)

# Add gene name, strand, editing type.
mcols(gr) = data.frame(mcols(gr), founders)

# Determine the editing type by getting the two highest
# expressed alleles.
for(i in 1:length(gr)) {
  ed = matrix(unlist(mcols(gr)[i,-(1:4)]), nrow = 4, dimnames =
       list(c("A", "C", "G", "T"), NULL))
  ed = rowSums(ed)
  ed = names(ed)[order(ed)][3:4]
  ed = ed[!ed %in% gr$ref[i]]
  gr$edit.type[i] = paste0(gr$ref[i], ed)
} # for(i)

# Remove genes with more than one non-canonical editing site per gene.
# We beleive that mulitple non-canonical editing sites are alignment
artifacts.
non.canon = gr$edit.type != "AG" & gr$edit.type != "CT"
spl = split(non.canon, gr$gene_id)
spl = sapply(spl, sum)

# Keep genes with 0 non-canonical editing sites.
keep = names(spl)[spl <= 0]
gr = gr[gr$gene_id %in% keep,]

# Now get rid of the non-canonical editing sites.
length(gr)
gr = gr[gr$edit.type %in% c("AG", "CT")]
length(gr)

# Make sure that all strains have non-canonical counts < 1% of total.
keep = rep(F, length(gr))
tmp = as.matrix(mcols(gr)[,-(1:4)])
```

```
counts = matrix(0, length(gr), 2, dimnames = list(gr$symbol, c("AG",
"CT")))
for(i in 1:length(gr)) {
  ag.sum = tmp[i,seq(1, ncol(tmp), 4)] + tmp[i,seq(3, ncol(tmp), 4)]
  ct.sum = tmp[i,seq(2, ncol(tmp), 4)] + tmp[i,seq(4, ncol(tmp), 4)]
  counts[i,] = c(sum(ag.sum), sum(ct.sum))
  if(gr$edit.type[i] == "AG") {
    keep[i] = all(ct.sum / (ag.sum + ct.sum) < 0.01, na.rm = TRUE)
  } else if(gr$edit.type[i] == "CT") {
    keep[i] = all(ag.sum / (ag.sum + ct.sum) < 0.01, na.rm = TRUE)
  } # else if(gr$edit.type[i] == "CT")
} # for(i)

length(gr)
gr = gr[keep]
length(gr)

print(paste("Num. sites:", length(gr)))
print(paste("Num. genes:", length(unique(gr$gene_id))))
print(table(gr$edit.type))

# This file contains the putative editing sites in the DO founders.
write.csv(as.data.frame(gr), file = "_5_founder_editing_union.csv")
```

File S7

Pile up denovo sites in DO

```
#########################################################################
##
# Pile up the denovo editing sites found in the founders in the DO.
# Daniel Gatti
# dan.gatti@jax.org
# Sept. 5, 2015
#########################################################################
##
library(Rsamtools)
library(GenomicRanges)
options(stringsAsFactors = F)
setwd("/hpcdata/dgatti/RNAediting/")

# Read in the denovo editing sites from the founders.
sites = read.csv("founders/_5_founder_editing_union.csv")
sites = GRanges(seqnames = sites$seqnames, range = IRanges(start =
        sites$start, width = 1), strand = sites$strand, mcols =
        sites[,-(1:6)])
colnames(mcols(sites)) = sub("^mcols\\.", "", colnames(mcols(sites)))
names(sites) = paste(seqnames(sites), start(sites), sep = "_")

# Get the DO BAM files in Anuj's directory.
bamdir =
"/hpcdata/anuj/Projects/Investigator/Gary_ChurChill/RNA_editing/DO_Ana
lysis/Analysis_Dir"
bamfiles = dir(path = bamdir, pattern = "bam$", recursive = T,
full.names = T)
bamfiles = bamfiles[-grep("test", bamfiles)]
stopifnot(length(bamfiles) == 277)

# Create PileupFiles.
pufiles = PileupFiles(bamfiles)

# Make the pileup function.
pufxn = function(x) {
          dimnames(x$seq)[[3]] = x$pos
          x$seq
        } # pufxn()

param = ApplyPileupsParam(minBaseQual = 20L, maxDepth = 1e7,
        what = "seq", which = sites)

# Pileup all 277 samples at each edit site.
pileup = applyPileups(files = pufiles, FUN = pufxn, param = param)
names(pileup) = names(sites)

save(pileup, file = "DO/DO_pileup_denovo_sites.Rdata")
```

```
print(paste(length(pileup), "sites in DO."))

# Keep sites with mean coverage >= 20.
coverage = sapply(pileup, colSums)
keep = which(colMeans(coverage) >= 20)
pileup = pileup[keep]
sites  = sites[keep]

print(paste(length(pileup), "sites with coverage >= 20 in DO."))

compl = c("A", "C", "G", "T")
names(compl) = c("T", "G", "C", "A")

# Keep sites with mean edit ratio >= 0.02.
keep = rep(FALSE, length(pileup))
for(i in 1:length(pileup)) {

  if(as.character(strand(sites)[i]) == "+") {

    ref = sites$ref[i]
    alt = sub(sites$ref[i], "", sites$edit.type[i])

  } else {

    ref = compl[sites$ref[i]]
    alt = compl[sub(sites$ref[i], "", sites$edit.type[i])]

  } # else

  er = mean(pileup[[i]][alt,,1] / (pileup[[i]][ref,,1] +
       pileup[[i]][alt,,1]), na.rm = T)
  keep[i] = er >= 0.02

} # for(i)

print(paste(sum(keep), "sites with mean edit ratio > 0.02 of",
      length(sites), "sites"))

sites  = sites[keep]
pileup = pileup[keep]

save(sites, pileup, file = "DO/_6_denovo_edit_sites.Rdata")
```

File S8

Remove sites that occur predominantly at the ends of reads

```
#########################################################################
##########
# For each of the discovered editing sites, record the bp position in
each
# read in the founders.
#
# Daniel Gatti
# dan.gatti@jax.org
# Aug. 11, 2015
# Ensembl 68
# Sanger v3
#########################################################################
##########
library(GenomicRanges)
library(GenomicAlignments)
library(Rsamtools)
library(foreach)
library(doParallel)
library(BSgenome.Mmusculus.UCSC.mm10)
mm10 = BSgenome.Mmusculus.UCSC.mm10

setwd("/hpcdata/dgatti/RNAediting/")

# Read in the denovo founder editing sites.
load(file = "DO/_6_denovo_edit_sites.Rdata")

# Get the founder BAM files in Anuj's directory.
bamdir =
"/hpcdata/anuj/Projects/Investigator/Gary_ChurChill/RNA_editing/Final_
DataSet/FounderBams"
bamfiles = dir(path = bamdir, pattern = "bam$", full.names = T)

cl = makeCluster(10)
registerDoParallel(cl)

readpos = foreach(bf = iter(bamfiles),
                  .packages = c("Rsamtools", "GenomicRanges",
"GenomicAlignments"),
                  .export = "sites") %dopar% {

  # Get the alignments from the BAM.
  param = ScanBamParam(which = sites)
  ga = readGAlignments(BamFile(bf), use.names = TRUE, param = param,
       with.which_label = TRUE)
  mcols(ga)$which_label = sub("\\-[0-9]+$", "", mcols(ga)$which_label)
  mcols(ga)$which_label = sub(":", "_", mcols(ga)$which_label)
  names = as.character(runValue(mcols(ga)$which_label))
```

```
  ga = split(x = ga, f = mcols(ga)["which_label"])
  names(ga) = names
  retval = vector("list", length(ga))
  names(retval) = names(ga)

  compl = c("A", "C", "G", "T", "N")
  names(compl) = c("T", "G", "C", "A", "N")

  # Loop through the reads that align to each site.
  # We only have M, D, I and N in these files. Widths range from 100+.
  for(i in 1:length(ga)) {

    curr.site = sites[names(ga)[i]]
    site.pos = rep(0, length(ga[[i]]))
    strand = as.character(strand(curr.site))
    ref = curr.site$ref
    alt = sub(ref, "", curr.site$edit.type)

    # Get the bp calls for the reads.
    param = ScanBamParam(what = c("qname", "strand", "pos", "cigar",
"qwidth",  "seq"),
             which = curr.site)
    reads = scanBam(file = bf, param = param)[[1]]

    # Get the read locations on reference coordinates. This should
    # parse up the reads using the CIGAR string.
    tmp = extractAlignmentRangesOnReference(cigar = reads$cigar, pos =
reads$pos)

    ops = explodeCigarOps(reads$cigar)
    len = explodeCigarOpLengths(reads$cigar)
    optypes = colSums(cigarOpTable(reads$cigar))

    # If we have only "M", then skip the special processing.
    if(sum(optypes[2:9]) == 0) {

      site.pos = start(curr.site) - as.integer(start(tmp)) + 1

    } else {

      for(j in 1:length(tmp)) {

        # When there is a "D", the alignment isn't broken up.
        if(length(grep("D", ops[[j]])) > 0) {

           st = c(reads$pos[j], reads$pos[j] + sum(len[[j]][1:2]))
           en = c(reads$pos[j] + len[[j]][1] - 1, reads$pos[j] +
sum(len[[j]][1:3]) - 1)
           tmp[[j]] = IRanges(start = st, end = en)

        } # if(length(grep("D", ops[[j]])) > 0)
```

```
        ol = findOverlaps(tmp[[j]], ranges(curr.site))
        hit = queryHits(ol)

        # Not sure why scanBam() is returning reads that don't overlap
the
        # editing sites...
        if(length(hit) > 0) {
          site.pos[j] = start(curr.site) - start(tmp[[j]])[hit] + 1
          if(hit > 1) {
            site.pos[j] = site.pos[j] + sum(width(tmp[[j]])[1:(hit-
1)])
          } # if(queryHits(ol) > 1)
        } # if(length(hit) > 0)

      } # for(j)

    } # else

    keep = which(site.pos > 0)
    allele = as.character(subseq(reads$seq[keep], start =
site.pos[keep],
             width = 1))
    if(strand == "-") {
      allele = compl[allele]
    } # if(strand == "-")

    if(any(!allele %in% c(ref,alt))) {
      print(paste(i, "Expected", paste(ref, alt, sep = ","),
"Observed",
            paste(unique(allele), collapse = ",")))
    } # if(any(!allele %in% c(ref,alt)))

    # Save the positions and put the allele calls in the names of
retval.
    retval[[i]] = site.pos[keep]
    names(retval[[i]]) = allele

  } # for(i)

  retval

} # for(bf)

stopCluster(cl)

names(readpos) =
sub("^/hpcdata/anuj/Projects/Investigator/Gary_ChurChill/RNA_editing/F
inal_DataSet/FounderBams/",
                "", bamfiles)
save(readpos, file = "founders/_7_edit_site_read_position.Rdata")

###
```

```
# Load in 'readpos'.
load(file = "founders/_7_edit_site_read_position.Rdata")

# Compile the read positions in each of the either founders for each
of
# the 192 sites.
strains = substring(names(readpos), 1, 3)
readpos = split(readpos, strains)
ref.pos.by.founder = vector("list", length(readpos))
names(ref.pos.by.founder) = names(readpos)
alt.pos.by.founder = vector("list", length(readpos))
names(alt.pos.by.founder) = names(readpos)

for(i in 1:length(readpos)) {

  ref.pos = matrix(0, nrow = length(sites), ncol = 100, dimnames =
            list(names(sites), 1:100))
  alt.pos = matrix(0, nrow = length(sites), ncol = 100, dimnames =
            dimnames(ref.pos))

  # Loop through each replicate and sum the positions for each site.
  for(s in 1:length(readpos[[i]])) {

    m = match(names(readpos[[i]][[s]]), rownames(ref.pos))
    ref = sites[names(readpos[[i]][[s]])]$ref

    for(j in 1:length(readpos[[i]][[s]])) {

      alt = sub(sites[names(readpos[[i]][[s]][j])]$ref, "",
            sites[names(readpos[[i]][[s]][j])]$edit.type)
      fac = factor(readpos[[i]][[s]][[j]], levels = 1:100)
      tbl = table(fac[names(fac) == ref[j]])
      ref.pos[m[j],] = ref.pos[m[j],] + tbl
      tbl = table(fac[names(fac) == alt])
      alt.pos[m[j],] = alt.pos[m[j],] + tbl

    } # for(j)
  } # for(s)

  ref.pos.by.founder[[i]] = ref.pos
  alt.pos.by.founder[[i]] = alt.pos

} # for(i)

pos.by.founder = ref.pos.by.founder
for(i in 1:length(pos.by.founder)) {
  pos.by.founder[[i]] = pos.by.founder[[i]] + alt.pos.by.founder[[i]]
} # for(i)

pdf("edit_pos_in_reads.pdf", width = 10, height = 8)
for(i in 1:nrow(pos.by.founder[[1]])) {
  layout(matrix(1:8, 2, 4, byrow = TRUE))
```

```
  par(plt = c(0.1, 0.9, 0.1, 0.88))
  for(j in 1:length(pos.by.founder)) {
    plot(pos.by.founder[[j]][i,], main =
paste(names(pos.by.founder)[j],
        rownames(pos.by.founder[[j]])[i]), ylab = "", las = 1)
  } # for(j)
} # for(i)
dev.off()

# Filter the reads based on whether most of the edit site positions
# occur at this distal ends of the reads.
keep = rep(FALSE, nrow(pos.by.founder[[1]]))
names(keep) = rownames(pos.by.founder[[1]])

# For each site....
pv = matrix(NA, nrow(pos.by.founder[[i]]), 8, dimnames =
     list(rownames(pos.by.founder[[1]]), names(pos.by.founder)))
for(i in 1:length(pos.by.founder)) {

  tot = rowSums(pos.by.founder[[i]])
  ends = rowSums(pos.by.founder[[i]][,c(1:5, 95:100)])
  for(j in 1:nrow(pos.by.founder[[i]])) {
   if(tot[j] > 0) {
     tst = binom.test(x = ends[j], n = tot[j], p = 0.1,
           alternative = "greater", conf.level = 0.8)
     pv[j,i] = tst$p.value
   } # if(tot[j] > 0)
  } # for(j)

} # for(i)

# Apply a Holm correction to the p-values.
pv.adj = matrix(p.adjust(pv, method = "holm"), nrow(pv),
         dimnames = dimnames(pv))

# If any site has an adjusted p-value < 0.05, remove it.
keep = which(apply(pv.adj, 1, min) >= 0.05)
remove = which(apply(pv.adj, 1, min) < 0.05)

discarded.sites = sites[remove]
sites = sites[keep]

print(paste("Discarded", length(discarded.sites), "of", nrow(pv.adj)))
print(paste("Keeping", length(sites), "of", nrow(pv.adj)))

save(sites, file =
"founders/_8_denovo_editing_sites_read_pos_filtered.Rdata")
write.csv(as.data.frame(discarded.sites),
"founders/_8_denovo_editing_sites_discarded.csv")
```

File S9

RADAR and DARNED pileup in DO

```
#########################################################################
##########
# Pileup the DARNED and RADAR sites in the DO.
# PileupParams: minBaseQual = 20
#               maxDepth = 1e7
#
# Daniel Gatti
# dan.gatti@jax.org
# Aug. 11, 2015
# Ensembl 68
# Sanger v3
#########################################################################
##########
options(stringsAsFactors = FALSE)
library(GenomicRanges)
library(GenomicAlignments)
library(Rsamtools)
library(rtracklayer)
library(foreach)
library(doParallel)
library(DOQTL)

# Load in the ensembl 68 GTF.
load("/hpcdata/cgd/ensembl/release68/Mus_musculus.GRCm38.68.Rdata")

# Keep the autosomes, X, Y and M.
ensembl = keepSeqlevels(x = ensembl, value =
          seqlevels(ensembl)[-grep("JH|GL", seqlevels(ensembl))])

# Keep only unique exons and UTRs.
length(unique(ensembl$gene_id))
ensembl = ensembl[ensembl$feature == "exon"]
keys = paste0(seqnames(ensembl), start(ensembl), end(ensembl))
ensembl = ensembl[!duplicated(keys)]
length(unique(ensembl$gene_id))

setwd("/hpcdata/dgatti/RNAediting/")

# First read in the RADAR sites (which are on mm9 for some reason) and
lift
# them over to mm10.
radar.mm9 = read.delim("RADAR_DARNED/Mouse_AG_all_mm9_RADARv2.txt")
radar.mm9 = GRanges(seqnames = radar.mm9$chromosome, ranges =
IRanges(start =
            radar.mm9$position, width = 1), strand = radar.mm9$strand,
            gene = radar.mm9$gene, inchr = "A", inrna = "I")
chain = import.chain("RADAR_DARNED/mm9ToMm10.over.chain")
radar.mm10 = liftOver(radar.mm9, chain)
radar.mm10 = stack(radar.mm10)
```

```
mcols(radar.mm10) = mcols(radar.mm10)[colnames(mcols(radar.mm10)) !=
"sample"]
rm(radar.mm9)

# Read in the DARNED data (whcih is on mm10 coordinates) and combine
with
# the RADAR data.
darned = read.delim("RADAR_DARNED/darned_mm10.txt")
darned = darned[(darned$inchr == "C" & darned$inrna == "U") |
                (darned$inchr == "A" & darned$inrna == "I"),]
darned = GRanges(seqnames = paste0("chr", darned$chrom), ranges =
         IRanges(start = darned$coordinate, width = 1), strand =
darned$strand,
         gene = darned$gene, inchr = darned$inchr, inrna =
darned$inrna)

sites = GRangesList(radar.mm10, darned)
sites = stack(sites)
sites = sites[order(start(sites))]
sites = sites[order(seqnames(sites))]
sites = sites[!duplicated(paste0(seqnames(sites), start(sites)))]
names(sites) = paste(sites$gene, as.character(seqnames(sites)),
                     start(sites), sep = "_")
length(sites)
save(sites, file = "RADAR_DARNED/radar_darned_sites_combined.Rdata")

# Get the DO BAM files in Anuj's directory.
bamdir =
"/hpcdata/anuj/Projects/Investigator/Gary_ChurChill/RNA_editing/DO_Ana
lysis/Analysis_Dir"
bamfiles = dir(path = bamdir, pattern = "bam$", recursive = T,
full.names = T)
bamfiles = bamfiles[-grep("test", bamfiles)]
stopifnot(length(bamfiles) == 277)

# Create PileupFiles.
pufiles = PileupFiles(bamfiles)

# Make the pileup function.
pufxn = function(x) {
          dimnames(x$seq)[[3]] = x$pos
          x$seq
        } # pufxn()

param = ApplyPileupsParam(minBaseQual = 20L, maxDepth = 1e7,
        what = "seq", which = sites)

# Pileup all 277 samples at each edit site.
pileup = applyPileups(files = pufiles, FUN = pufxn, param = param)
names(pileup) = names(sites)

save(pileup, file = "DO/DO_pileup_radar_darned.Rdata")
```

```
######
# Loads in 'sites'.
load(file = "RADAR_DARNED/radar_darned_sites_combined.Rdata")
radar.sites = sites
load(file = "DO/DO_pileup_radar_darned.Rdata")
radar.pileup = pileup

# Sum the reads across all DO samples.
alleles  = sapply(radar.pileup, rowSums)
coverage = sapply(radar.pileup, colSums)
mean.coverage = sapply(coverage, mean)
mean.coverage[is.nan(mean.coverage)] = 0

# Filter the sites to keep only those with > 0 reads.
print(paste("Total sites:", ncol(alleles)))
print(paste("Sites with 0 reads:", sum(colSums(alleles) == 0)))

alleles = alleles[,mean.coverage > 0]
mean.coverage = mean.coverage[mean.coverage > 0]
print(paste("Sites with > 0 reads:", ncol(alleles)))

alleles = alleles[,mean.coverage > 20]
mean.coverage = mean.coverage[mean.coverage > 20]
print(paste("Sites with > 20 reads:", ncol(alleles)))

# Verify that the sites that we retain have the same
# editing type as the databases.
radar.sites = radar.sites[colnames(alleles)]
radar.pileup = radar.pileup[colnames(alleles)]
stopifnot(names(radar.sites) == colnames(alleles))

# Sort the counts for each site.
sorted.alleles = split(t(alleles), colnames(alleles))
sorted.alleles = sorted.alleles[names(radar.sites)]
sorted.alleles = lapply(sorted.alleles, function(z) {
                   names(z) = rownames(alleles); z })
sorted.alleles = lapply(sorted.alleles, sort, decreasing = TRUE)
sorted.alleles.names = sapply(sorted.alleles, names)
stopifnot(names(sorted.alleles) == names(radar.sites))

# Get the expected and observed alleles.
tmp = data.frame(expchr = radar.sites$inchr, exprna =
radar.sites$inrna,
      strand = as.character(strand(radar.sites)),
      major = sorted.alleles.names[1,], minor =
sorted.alleles.names[2,],
      stringsAsFactors = FALSE)

# Change the I to G.
tmp$exprna[tmp$exprna == "I"] = "G"
```

```
# Change the U to T.
tmp$exprna[tmp$exprna == "U"] = "T"

# Get the complement of the alleles on the - strand.
compl = c("A", "C", "G", "T")
names(compl) = c("T", "G", "C", "A")
minus = which(tmp$strand == "-")
tmp$expchr[minus] = compl[tmp$expchr[minus]]
tmp$exprna[minus] = compl[tmp$exprna[minus]]

# Keep only the sites where the DO observed alleles match the
# expected alleles.
keep = which((tmp$expchr == tmp$major & tmp$exprna == tmp$minor) |
             (tmp$expchr == tmp$minor & tmp$exprna == tmp$major))
print(paste(nrow(tmp) - length(keep), "sites have different
alleles."))
print(paste(length(keep), "sites have the same alleles."))
radar.sites = radar.sites[keep]
alleles = alleles[,keep]
sorted.alleles = sorted.alleles[keep]

# Now verify that we don't have > 2% non-canonical reads.
total.cov = colSums(alleles)
noncanon  = colSums(sapply(sorted.alleles, "[", 3:5))
noncanon.ratio = noncanon / total.cov
keep = which(noncanon.ratio <= 0.02)
radar.sites = radar.sites[keep]
alleles = alleles[,keep]
sorted.alleles = sorted.alleles[keep]
print(paste(length(keep), "sites have <= 2% non-canonical reads."))

# Keep sites with a MAF >= 0.05.
maf = sapply(sorted.alleles, "[", 2) / colSums(sapply(sorted.alleles,
"[", 1:2))
keep = which(maf >= 0.05)
radar.sites = radar.sites[keep]
alleles = alleles[,keep]
sorted.alleles = sorted.alleles[keep]
radar.pileup = radar.pileup[names(radar.sites)]
print(paste(length(keep), "sites have >= 5% MAF."))

# Load in the DO edit sites and make sure that we don't already have
# these sites.
# Loads in 'sites'
load(file =
"founders/_7_denovo_editing_sites_read_pos_filtered.Rdata")
denovo.sites  = sites
denovo.pileup = pileup
rm(sites, pileup)

site.ol = findOverlaps(query = radar.sites, subject = denovo.sites)
print(paste("Sites already mapped =", length(site.ol)))
```

```
# Remove the sites that we already mapped.
remove = queryHits(site.ol)
radar.sites  = radar.sites[-remove]
radar.pileup = radar.pileup[names(radar.sites)]
print(paste("Sites from DB to map = ", length(radar.sites)))

# Write out the data that we have.
save(denovo.sites, radar.sites, denovo.pileup, radar.pileup,
     file = "DO/_8_denovo_radar_darned_sites.Rdata")


# Pile up the reads in the founders and record this.
bamdir =
"/hpcdata/anuj/Projects/Investigator/Gary_ChurChill/RNA_editing/Final_
DataSet/FounderBams"
bamfiles = dir(path = bamdir, pattern = "bam$", full.names = T)

# Make a list of radar.pileupFiles by strain.
pufiles = PileupFiles(bamfiles)

# Make the pileup function.
pufxn = function(x) {
          dimnames(x$seq)[[3]] = x$pos
          x$seq
        } # pufxn()

param = ApplyPileupsParam(minBaseQual = 20L, maxDepth = 1e7,
        what = "seq", which = radar.sites)

# Pileup all samples.
founder.pileup = applyPileups(files = pufiles, FUN = pufxn, param =
param)
names(founder.pileup) = names(radar.sites)

save(founder.pileup, file =
paste0("RADAR_DARNED/_8_radar_darned_founder_pileup.Rdata"))
```

File S10

Combine the denovo, RADAR and DARNED sites and prepare QTL data

```
#########################################################################
##########
# Merge the denovo and RADAR/DARNED edit sites into one table.
# Daniel Gatti
# dan.gatti@jax.org
# Sept. 4, 2015
#########################################################################
##########
options(stringsAsFactors = F)
library(DOQTL)
setwd("/hpcdata/dgatti/RNAediting")

# Load in the ensembl 68 GTF.
load("/hpcdata/cgd/ensembl/release68/Mus_musculus.GRCm38.68.Rdata")

# Add 'chr' to the chromosome names to match the BAM files.
ensembl = keepSeqlevels(x = ensembl, value =
          seqlevels(ensembl)[-grep("JH|GL", seqlevels(ensembl))])
ensembl = renameSeqlevels(x = ensembl, value = paste0("chr",
seqlevels(ensembl)))
sl = sub("MT", "M", seqlevels(ensembl))
ensembl = renameSeqlevels(x = ensembl, value = sl)

# Keep only unique exons and UTRs.
length(unique(ensembl$gene_id))
ensembl = ensembl[ensembl$feature == "exon"]
keys = paste0(seqnames(ensembl), start(ensembl), end(ensembl))
ensembl = ensembl[!duplicated(keys)]
length(unique(ensembl$gene_id))

load(file = "RADAR_DARNED/_8_radar_darned_founder_pileup.Rdata")
load(file = "DO/_8_denovo_radar_darned_sites.Rdata")

# Create a summary file for the RADAR/DARNED sites like the one for
denovo sites.
strains = factor(substring(colnames(founder.pileup[[1]]), 1, 3))
site.pileup = matrix(0, length(founder.pileup), 4 *
length(levels(strains)),
             dimnames = list(names(founder.pileup),
             paste(rep(levels(strains), each = 4),
c("A","C","G","T"), sep = "_")))

for(i in 1:length(founder.pileup)) {

  agg = aggregate(data.frame(t(founder.pileup[[i]][,,1])),
list(strains), sum)
  agg = as.matrix(agg[,-c(1,6)])
  if(as.character(strand(radar.sites)[i]) == "-") {
```

```
    tmp = agg[,c(1,3)]
    agg[,c(1,3)] = agg[,c(2,4)]
    agg[,c(2,4)] = tmp

  } # if(as.character(strand(radar.sites)[i]) == "-")

  site.pileup[i,] = as.vector(t(agg))

} # for(i)

# Combine the data and make the column names the same as the denovo
sites
radar.sites = cbind(as.data.frame(radar.sites), site.pileup)
colnames(radar.sites) = sub("inchr", "ref", colnames(radar.sites))
colnames(radar.sites) = sub("inrna", "edit.type",
colnames(radar.sites))
colnames(radar.sites) = sub("gene", "symbol", colnames(radar.sites))
colnames(radar.sites) = sub("sample", "gene_id",
colnames(radar.sites))
radar.sites$edit.type[radar.sites$edit.type == "U"] = "T"
radar.sites$edit.type[radar.sites$edit.type == "I"] = "G"
radar.sites$edit.type = paste0(radar.sites$ref, radar.sites$edit.type)
ensid = ensembl$gene_id[match(radar.sites$symbol, ensembl$gene_name)]
radar.sites$gene_id = ensid

radar.sites = GRanges(seqnames = radar.sites$seqnames, range =
IRanges(
              start = radar.sites$start, width = 1), strand =
radar.sites$strand,
              mcols = radar.sites[,-(1:5)])
colnames(mcols(radar.sites)) = sub("^mcols\\.", "",
colnames(mcols(radar.sites)))

write.csv(as.data.frame(radar.sites), file =
"RADAR_DARNED/_9_darned_radar_edit_sites.csv",
          quote = FALSE)

colnames(mcols(radar.sites)) = colnames(mcols(denovo.sites))
sites  = c(denovo.sites, radar.sites)
pileup = c(denovo.pileup, radar.pileup)

sites$source = rep(c("denovo", "radar"), c(length(denovo.sites),
length(radar.sites)))
names(sites) = paste(seqnames(sites), start(sites), sep = "_")

write.csv(as.data.frame(sites), file =
"_9_all_edit_sites_for_mapping.csv",
          quote = F)
save(sites, pileup, file = "_9_all_edit_sites.Rdata")

# Gather the data for QTL mapping.
```

```
edit  = matrix(0, ncol(pileup[[1]]), length(sites), dimnames = list(
        colnames(pileup[[1]]), names(sites)))
rownames(edit) = sub("\\.final\\.sorted\\.bam", "", rownames(edit))
total = matrix(0, nrow(edit), ncol(edit), dimnames = dimnames(edit))
compl = c("A", "C", "G", "T")
names(compl) = c("T", "G", "C", "A")
for(i in 1:length(sites)) {

  ref = sites$ref[i]
  alt = sub(ref, "", sites$edit.type[i])

  if(as.character(strand(sites)[i]) == "-") {
    ref = compl[ref]
    alt = compl[alt]
  } # if(as.character(strand(sites)[i]) == "-")

  edit[i,]  = pileup[[i]][alt,,1]
  total[i,] = pileup[[i]][alt,,1] + pileup[[i]][ref,,1]

} # for(i)

# Read in the genotype probabilities and assemble the mapping data.
load("/hpcdata/cgd/DO_genoprobs/MUGA_founder_probs_v2.Rdata")
probs = model.probs[grep("^KLS", rownames(model.probs)),,]
rm(model.probs)
rownames(probs) = sub("^KLS", "", rownames(probs))
probs = probs[rownames(edit),,]

stopifnot(all(rownames(edit) == rownames(probs)))

# Read in the MUGA markers.
load(url("ftp://ftp.jax.org/MUGA/muga_snps.Rdata"))
snps = muga_snps[muga_snps[,1] %in% dimnames(probs)[[3]],]
probs = probs[,,snps[,1]]

# Create a list of kinship matrices.
K = kinship.probs(probs = probs, snps = snps, bychr = T)

# Load in the sex and diet covariates.
covar = read.csv("QTL/svenson_277_covar.csv")
rownames(covar) = covar[,1]
covar = as.matrix(covar[,-1])
covar = covar[rownames(edit),]

covar = covar[rownames(probs),]
total = total[rownames(probs),]
edit  = edit[rownames(probs),]
probs = probs[rownames(probs),,]

stopifnot(nrow(probs) == nrow(total))
stopifnot(all(rownames(probs) == rownames(total)))
stopifnot(nrow(probs) == nrow(edit))
```

```
stopifnot(all(rownames(probs) == rownames(edit)))
stopifnot(all(rownames(K[[1]]) == rownames(total)))

# Save this data for mapping.
save(sites, total, edit, covar, probs, K, covar, snps,
     file = "QTL/_9_RNAediting_QTL_mapping.Rdata")
```

File S11

QTL mapping of editing ratios

```
#########################################################################
##########
# Read in the RNA editing sites in the DO and map them.
# Daniel Gatti
# Dan.Gatti@jax.org
# Nov. 19, 2014
#########################################################################
##########
library(DOQTL)
setwd("/hpcdata/dgatti/RNAediting/QTL/")

args = commandArgs(trailingOnly = T)

# Load in the mapping data.
load("_9_RNAediting_QTL_mapping.Rdata")

i = as.numeric(args)

print(date())

# Set low denominator samples = 0.
total[total < 10] = NA

# Map editing counts with total counts as a covariate.
pheno = matrix(edit[,i], ncol = 1, dimnames = list(rownames(edit),
        colnames(edit)[i]))
addcovar = cbind(covar, total = total[,i])
addcovar[addcovar[,3] == 0,] = NA

qtl = scanone(pheno = pheno, pheno.col = 1, probs = probs, K = K,
      addcovar = addcovar, snps = snps)

perms = scanone.perm(pheno = pheno, pheno.col = 1, probs = probs,
        addcovar = addcovar, snps = snps)

save(qtl, perms, file =
paste0(colnames(edit)[i],"_RNAediting_QTL.Rdata"))

print(date())
```

File S12

Harvest maximum QTL peaks

```
#########################################################################
##########
# Harvest the QTL by taking the maximum peak and determining if it it
over
# the 0.05 treshold.
# Daniel Gatti
# dan.gati@jax.org
# Sept. 14, 2015
#########################################################################
##########
options(stringsAsFactors = FALSE)
library(DOQTL)

setwd("/hpcdata/dgatti/RNAediting/QTL/")

load("_9_RNAediting_QTL_mapping.Rdata")

files = dir(pattern = "_RNAediting_QTL.Rdata$")

# Get the chromosome lengths.
chrlen = get.chr.lengths()
chrlen = c(0, chrlen)

result = NULL

for(i in 1:length(files)) {

  # Get the site.
  site = sites[sub("_RNAediting_QTL.Rdata$", "", files[i])]

  # Add the DO editing ratio to the end of the site.
  site$DO.total = mean(total[,names(site)], na.rm = TRUE)
  site$DO.ratio = mean(edit[,names(site)] / total[,names(site)], na.rm
= TRUE)

  # Load in the QTL and perms.
  load(files[i])
  # Get the 0.05 threshold.
  thr = quantile(perms, 0.95)

  # Plot the genome scan.
  png(sub("\\.Rdata$", ".png", files[i]), width = 1000, height = 800,
res = 128)
  plot(qtl, sig.thr = thr, main = names(site))
  site.chr = as.character(sub("^chr", "", seqnames(site)))
  if(site.chr == "X") {
    site.pos = start(site) * 1e-6 + sum(chrlen[1:20])
```

```
  } else if(site.chr == "Y") {
    site.pos = -100
  } else {
    site.pos = start(site) * 1e-6 +
sum(chrlen[1:as.numeric(site.chr)])
  } # else
  points(site.pos, 0, pch = 17, cex = 2, col = 2)
  dev.off()

  spl = split(qtl$lod$A, qtl$lod$A[,2])
  spl$X = qtl$lod$X
  max.lod = lapply(spl, function(z) { z[which.max(z[,7]),] })
  max.lod = unsplit(max.lod, names(spl))

  # Only keep cis-QTL on X.
  if(as.character(seqnames(site)) == "chrX") {
    if(max.lod[max.lod[,2] == "X",7] < 2 * thr) {
      max.lod = max.lod[max.lod[,2] != "X",]
    } # if(max.lod[max.lod[,2] == "X",7] < 2 * thr)
  } else if(as.character(seqnames(site)) == "chrY") {
    # Nothing
  } else {
    max.lod = max.lod[max.lod[,2] != "X",]
  } # else

#  max.lod = max.lod[max.lod[,7] >= thr,,drop = FALSE]
   max.lod = max.lod[which.max(max.lod[,7]),,drop = FALSE]
   max.lod = cbind(max.lod, p.gw = mean(perms >= max.lod[,7]))
   result = rbind(result, cbind(as.data.frame(site), max.lod))

   for(j in 1:nrow(max.lod)) {

    png(sub("_QTL\\.Rdata$", paste0("_coef_chr", max.lod[j,2],
".png"), files[i]),
        width = 1000, height = 800, res = 128)
    if(max.lod[j,2] == "X") {
#       coefplot(qtl, chr = max.lod[j,2], sex = "F", main =
names(site))
    } else {
      coefplot(qtl, chr = max.lod[j,2], main = names(site))
    } # else
    dev.off()

  } # for(j)

} # for(i)

result = cbind(result, p.adj = p.adjust(p = result$p.gw, method =
"BH"))

save(result, file = "_11_RNAediting_qtl_results.Rdata")
write.csv(result, file = "_11_RNAediting_qtl_results.csv")
```

File S13

Calculating the probability that the editing site occurs in a dsRNA region

```
#####Input Data #####

#P – Base pairing probability matrix as returned by RNAfold –p

#uP – Array with the probabilities of each nucleotide of being unpaired

#pos – Position of the edit site

#n – Length of the RNA sequence for the whole gene

###################


def probOfPosFavorable(P,uP,pos,n):
 # Prob of stem:

 sum = 0.0
 for i in range(1,n+1):
  sum += P[pos][i]

 punp = 1.0-sum

 #Left Bulge
 sumLB = 0.0
 for i in range(pos+7,n):
  sumLB += P[pos+1][i]*P[pos-1][i+1]
 for i in range(pos+8,n):
  sumLB += P[pos+2][i]*P[pos-1][i+1]*uP[pos+1]
 for i in range(pos+7,n):
  sumLB += P[pos+1][i]*P[pos-2][i+1]*uP[pos-1,n]

 #Right Bulge
 sumRB = 0.0
 for i in range(pos-7,2,-1):
  sumRB += P[pos-1][i]*P[pos+1][i-1]
 for i in range(pos-8,2,-1):
  sumRB += P[pos-2][i]*P[pos+1][i-1]*uP[pos-1]
 for i in range(pos-7,2,-1):
  sumRB += P[pos-1][i]*P[pos+2][i-1]*uP[pos+1]
```

```
#Left IL
sumLIL = 0.0
for i in range(pos+7,n-1):
  sumLIL += P[pos+1][i]*P[pos-1][i+2]*uP[i+1]
for i in range(pos+8,n-1): #2x1 a
   sumLIL += P[pos+2][i]*P[pos-1][i+2]*uP[pos+1]*uP[i+1]
for i in range(pos+7,n-1): #2x1 b
   sumLIL += P[pos+1][i]*P[pos-2][i+2]*uP[pos-1]*uP[i+1]
for i in range(pos+8,n-2): #2x2 a
   sumLIL += P[pos+2][i]*P[pos-1][i+3]*uP[pos+1]*uP[i+1]*uP[i+2]
for i in range(pos+7,n-2): #2x2 b
   sumLIL += P[pos+1][i]*P[pos-2][i+3]*uP[pos-1]*uP[i+1]*uP[i+2]
#Right IL
sumRIL = 0.0
for i in range(pos-7,3,-1):
  sumRIL += P[pos-1][i]*P[pos+1][i-2]*uP[i+1]
for i in range(pos-8,3,-1): #1x2 a
  sumLIL += P[pos-2][i]*P[pos+1][i-2]*uP[pos-1]*uP[i-1]
for i in range(pos-7,3,-1): #1x2

  sumLIL += P[pos-1][i]*P[pos+2][i-2]*uP[pos+1]*uP[i-1]
for i in range(pos-8,4,-1): #2x2 a
  sumLIL += P[pos-2][i]*P[pos+1][i-3]*uP[pos-1]*uP[i-1]*uP[i-2]
for i in range(pos-7,4,-1): #2x2 b
  sumLIL += P[pos-1][i]*P[pos+2][i-3]*uP[pos+1]*uP[i-1]*uP[i-2]

sum2 = punp*(sumLB+sumRB+sumLIL+sumRIL)


return sum + sum2
```