nature protocols

PROTOCOL

https://doi.org/10.1038/s41596-019-0279-7

Investigating RNA editing in deep transcriptome datasets with REDItools and REDIportal

Claudio Lo Giudice[1], Marco Antonio Tangaro[1], Graziano Pesole[1,2,3] and Ernesto Picardi[1,2,3*]

RNA editing is a widespread post-transcriptional mechanism able to modify transcripts through insertions/deletions or base substitutions. It is prominent in mammals, in which millions of adenosines are deaminated to inosines by members of the ADAR family of enzymes. A-to-I RNA editing has a plethora of biological functions, but its detection in large-scale transcriptome datasets is still an unsolved computational task. To this aim, we developed REDItools, the first software package devoted to the RNA editing profiling in RNA-sequencing (RNAseq) data. It has been successfully used in human transcriptomes, proving the tissue and cell type specificity of RNA editing as well as its pervasive nature. Outcomes from large-scale REDItools analyses on human RNAseq data have been collected in our specialized REDIportal database, containing more than 4.5 million events. Here we describe in detail two bioinformatic procedures based on our computational resources, REDItools and REDIportal. In the first procedure, we outline a workflow to detect RNA editing in the human cell line NA12878, for which transcriptome and whole genome data are available. In the second procedure, we show how to identify dysregulated editing at specific recoding sites in post-mortem brain samples of Huntington disease donors. On a 64-bit computer running Linux with ≥32 GB of random-access memory (RAM), both procedures should take ~76 h, using 4 to 24 cores. Our protocols have been designed to investigate RNA editing in different organisms with available transcriptomic and/or genomic reads. Scripts to complete both procedures and a docker image are available at https://github.com/BioinfoUNIBA/REDItools.

Introduction

Epitranscriptome refers to all biochemical RNA modifications affecting cellular RNAs[1]. In contrast to DNA modifications for which a role in gene expression has already been established, the biological significance of chemical alterations including isomerization and editing of canonical bases has not been well characterized. To date, >150 different RNA modifications have been discovered[2], and, among them, RNA editing represents the most prominent post-transcriptional mechanism[3]. First identified in 1986 in Trypanosoma mitochondria[4], RNA editing occurs in a wide range of organisms, altering primary transcripts by insertion/deletion or modification of specific nucleotides[5]. In mammals, it involves the deamination of cytosine to uridine or, more commonly, the conversion of adenosine to inosine (A-to-I) and, concurrently with alternative splicing, increases the diversity of the eukaryotic transcriptome and proteome[6]. While cytosine-to-uridine editing is carried out by APOBEC1 enzyme[7], and only a few instances have been discovered so far, A-to-I editing is pervasive and catalyzed by the adenosine deaminase family of enzymes acting on double stranded RNA[8–10].

RNA editing by base conversion can lead to a variety of biological effects depending on the RNA type (mRNA or non-coding RNA) or region (5′ or 3′ untranslated region (UTR), coding sequence (CDS) or intron) involved in the modification[5,11]. Accumulating evidence indicates that RNA editing changes in UTRs can alter gene expression and regulation, while modifications in coding protein regions can lead to amino acid replacements with several functional effects[11]. A-to-I RNA editing in mammals regulates innate immune response[12], and its deregulation has been implicated in the pathogenesis of many diseases including autoimmune and inflammatory tissue injury, neurodegenerative and psychiatric disorders and various tumors[13,14].

Nowadays, RNAseq is the de facto standard approach to discover RNA editing candidates in whole eukaryotic genomes[10,15]. Although the identification of editing sites is, in principle, quite simple, it represents a computational challenge because true RNA editing events have to be discriminated from genome-encoded single-nucleotide polymorphisms (SNPs) and technical artefacts caused by

[1]Institute of Biomembranes, Bioenergetics and Molecular Biotechnologies (IBIOM), National Research Council, Bari, Italy. [2]Department of Biosciences, Biotechnology and Biopharmaceutics, University of Bari, Bari, Italy. [3]National Institute of Biostructures and Biosystems (INBB), Rome, Italy.
*e-mail: ernesto.picardi@uniba.it

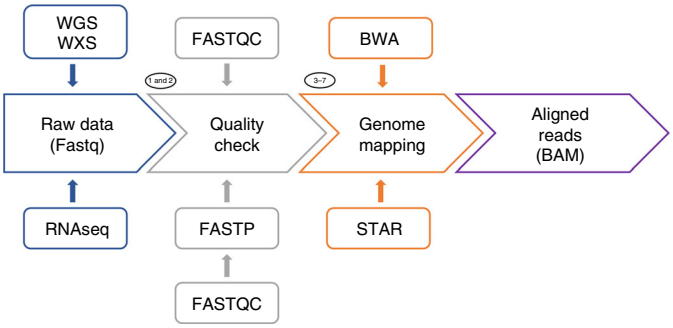

Fig. 1 | Overview of the bioinformatics workflow to preprocess data. Reliable RNA editing calls require good quality WGS and RNAseq reads. Once obtained from public databases, raw reads in fastq format are quality checked using FASTQC and cleaned using FASTP (Steps 1 and 2, Procedures 1 and 2). Then, RNAseq reads are aligned to the reference genome using a splice-aware software like STAR, while WGS reads are mapped using BWA (Steps 3–7, Procedures 1 and 2). Finally, aligned reads are converted into the standard BAM format for the downstream detection of RNA editing.

sequencing or read-mapping errors[10,15–19]. The use of genomic reads from whole genome sequencing (WGS) or whole exome sequencing (WXS) experiments in single individuals, annotations in the database of SNPs (dbSNP) and several stringent filters can minimize the detection of false RNA editing candidates[15–17].

To promote and facilitate the investigation of RNA editing at genomic scale, we developed and released the first software package devoted to this purpose, named REDItools[20]. It comprises a suite of scripts in the portable Python programming language and has been conceived to handle massive transcriptome sequencing data through a variety of filters to provide accurate RNA editing calls[20].

REDItools are routinely used worldwide to investigate A-to-I editing in mammalian genomes, and, to date, it has been applied in thousands of human RNAseq experiments, leading to the discovery of >4.6 million events in 55 human body sites, collected in our specialized REDIportal database[21].

REDItools and REDIportal represent two relevant and efficient computational resources to investigate RNA editing in a variety of organisms and in different physiological and pathological conditions. Here, we describe step-by-step two bioinformatics procedures to study A-to-I editing in human samples. In the first procedure, we illustrate a computational workflow to discover novel RNA editing events using RNAseq and WGS reads from the same sample. In the second procedure, we instead explore known RNA editing sites in RNAseq samples from donors affected by Huntington disease (HD) to identify potential dysregulated events. Although both procedures are based on human data in which A-to-I editing is abundant, they can be applied to other organisms as well.

Overview of the protocol

There are different ways to profile RNA editing in massive transcriptome datasets[17]. For simplicity, we describe here two procedures focusing on A-to-I RNA editing since it is widespread in mammals even if its functional roles are still largely unknown.

In the first procedure, we illustrate a workflow to identify A-to-I events in the human cell line NA12878, which is derived from a human sample belonging to the Centre d'Etude du Polymorphisme Humain (CEPH) population and has been deeply sequenced as part of the 1000 Genomes Project[22]. For this cell line, both the transcriptome and the genome have been sequenced using Illumina technology, and raw reads are freely available at the European Nucleotide Archive (ENA) (https://www.ebi.ac.uk/ena). The RNAseq data comprise 25,933,924 million raw paired-end reads (75 bases × 2) and have been generated by the Illumina HiSeq 2000 machine using a strand-oriented library. WGS data instead include 643,097,275 million raw paired-end reads (101 bases × 2) carried out by the same Illumina sequencer. Since input data files are very large, especially WGS files, the procedure detailed here is limited to the analysis of chromosome 4, making the workflow faster and easier for novice users. The entire procedure is human specific but can be applied to other organisms with available transcriptomic and genomic reads. The detection of RNA editing is carried out by our REDItools package, which is able to handle RNAseq data alone or combine RNAseq and genomic reads from WGS or WXS experiments to reduce the false discovery rate due to SNPs.

The first procedure begins with the download of RNAseq and WGS data in the standard fastq format (Fig. 1, stage 1) and the subsequent preprocessing to improve their global quality and ensure

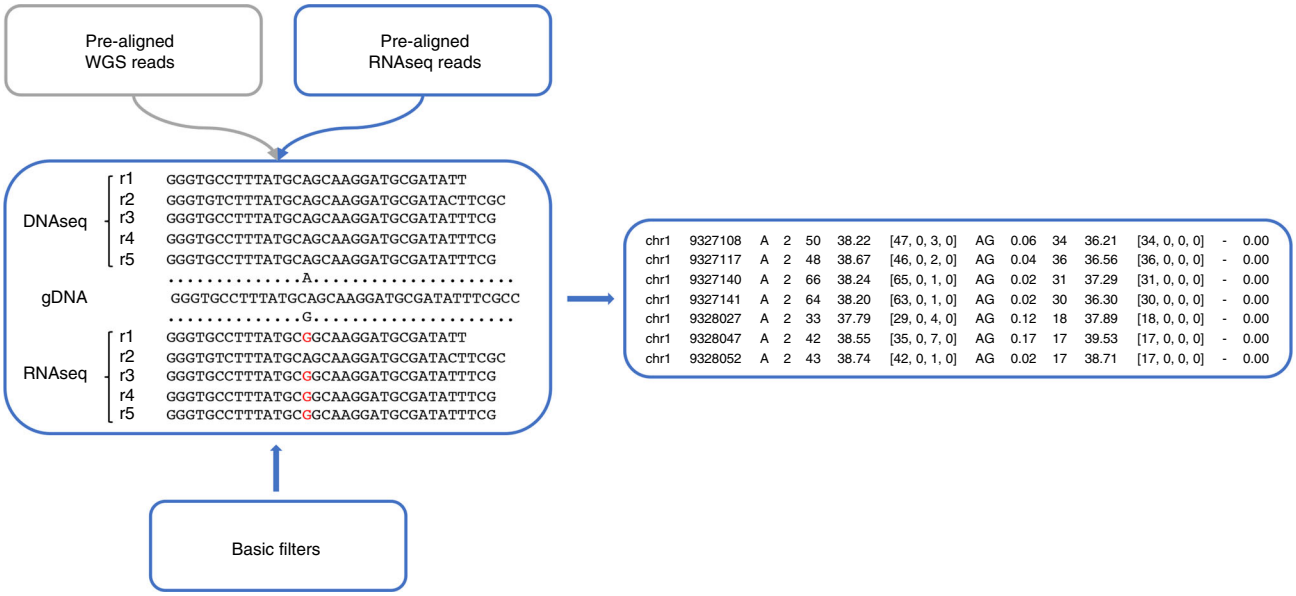

Fig. 2 | Detection of RNA editing by REDItools. Aligned RNAseq and WGS reads are passed to REDItools to identify all potential DNA–RNA variants. REDItools traverse RNAseq and WGS multiple read alignments position by position looking at RNA mismatches supported by WGS reads without SNP evidence. Several filters are applied to each position, and resulting variants are outputted in tab-delimited tables.

that input raw data are not biased (Fig. 1, stage 2). To this aim, collected reads are inspected using `FASTQC` (https://github.com/s-andrews/FastQC) to perform some quality control checks and cleaned using `FASTP`[23] to remove read regions of low quality or potential adaptor sequences or poly(A)-tails (or long terminal homopolymeric stretches) (Fig. 1, stage 2).

Next, cleaned reads are aligned onto the reference genome (Fig. 1, stage 3). While RNAseq reads are mapped using `STAR`[24], an ultrafast splice-aware software, WGS reads are aligned using `BWA`[25], which does not take into account the spliced nature of reads (Fig. 1, stage 3). Finally, aligned transcriptome and genome reads are converted in the standard BAM format using `SAMtools`[26] (Fig. 1, stage 4). Unlike `BWA`, `STAR` parameters can be tuned to directly output reads in BAM format, saving time and avoiding `SAMtools` calls.

After the preprocessing, RNAseq and WGS reads are passed to the `REDItoolDnaRna.py` script using non-stringent parameters to identify all potential DNA–RNA variants (Fig. 2). The use of loosing parameters, defined as basic filters in Fig. 2, is an expedient to save computing time in all cases in which a user needs to run multiple instances of REDItools with different option values. Indeed, several filters can be applied in post-processing using ad hoc auxiliary REDItools scripts as `FilterTable.py` or `selectPositions.py`. Filters implemented in REDItools are described in Box 1, whereas auxiliary REDItools scripts are detailed in Box 2.

At the end of the run, all DNA–RNA variants are returned in a simple and tab-delimited table (described in Box 3) and subjected to further filters according to the workflow depicted in Fig. 3. Briefly, positions from the first REDItools round are annotated by means of the `AnnotateTable.py` script using known SNP sites, repeated elements in RepeatMasker and editing events stored in our REDIportal database. Then, SNPs and sites not supported by ≥10 WGS reads are removed, and the remaining positions are divided according to RepeatMasker annotations in three groups: ALU, REP NON ALU and NON REP.

NON REP and REP NON ALU variants undergo a second round of `REDItoolDnaRna.py` using more stringent call criteria than their counterparts in ALU regions. In addition, reads supporting variants are collected and mapped onto the reference genome using PBLAT[27], a faster version of the classical BLAT tool[28], to detect reads mapping on multiple genome locations with similar scores. The same reads are also inspected to exclude PCR duplication. At the end, all filtered positions are collected, returning the final list of RNA editing candidates.

The second procedure focuses on the discovery of dysregulated recoding events in HD combining REDItools and REDIportal. HD is a fatal neurodegenerative disorder[29] caused by a CAG expansion in the gene encoding Huntingtin[29]. Its pathogenesis is characterized by altered gene expression as attested by several transcriptome studies on HD cells or brains harvested from mouse models or

Box 1 | REDItools filters

REDItools implements several positional filters to minimize biases due to sequencing and mapping errors. Below, we report a description of filters and default values, but users should tune them according to the characteristics of input RNAseq or WGS reads. In brackets, we also report command line options to active each filter in REDItoolsDnaRna.py script.

- Quality score: positions with a Phred score <25 are excluded (-q);
- Mapping quality: reads with a mapping quality score <30 are removed (-m in combination with -u for RNAseq and -U for WGS/WXS). This score is aligner-dependent and should be changed accordingly;
- Per base coverage: sites not supported by ≥10 reads are filtered out (-c);
- Bases supporting variation: sites with at least three reads supporting the variation are excluded (-v);
- Homopolymeric regions: positions in homopolymeric stretches of at least five bases are removed (-O in combination with -l for RNAseq and -L for WGS/WXS);
- Splice sites: positions near (i.e., within 4 bases) known splice sites are excluded (-V in combination with -w). This filter requires an input table with the list of known spice sites;
- Mis-mapping correction: positions in reads mapping to multiple genome locations are excluded (-b for RNAseq or -B for WGS/WXS). This filter requires an input list of mis-mapping reads from Blat or similar tools;
- Unique reads: only reads uniquely mapping are taken into account (-e for RNAseq and -E for WGS/WXS);
- Paired-end reads: only concordant reads are used (-p for RNAseq and -P for WGS/WXS). This filter works with paired-end reads only;
- PCR duplicates: reads marked as PCR duplicates are removed (-d for RNAseq and -D for WGS/WXS). This filter requires the preprocessing of the input BAM file with tools able to mark duplicate reads;
- Trimming: positions located a few bases upstream and/or downstream of single reads are excluded (-a for RNAseq and -A WGS/WXS);
- Infer strand: positions are handled according to the read strand (-s in combination with -g). This filter requires reads sequenced from a strand-oriented library. For reads that are not strand-oriented, the strand can be inferred using gene annotations provided in gtf format (-G);
- Strand correction: positions in opposite orientation to the detected strand are removed (-S in combination with -s);
- Editing level: sites showing editing levels lower than the background value of 0.1 are removed (-n);
- Multiple changes: positions showing multiple changes are excluded (-z for RNAseq and -Z for WGS/WXS);
- WGS/WXS support: positions not supported by WGS/WXS reads are discarded (-V);
- Invariant sites: positions without a reference mismatch are removed (-R);
- Selected changes: positions not showing a specific substitution type are removed (-W);
- Specific sites: positions not included in the user-provided list of sites are excluded (-T) or included (-K);
- Specific genomic region: positions not included in the user-provided genomic region are removed (-Y followed by the genomic region in the format chr:start-end).

postmortem HD brain tissues[30–33]. To check for disrupted RNA editing in HD, we use RNAseq data generated using the Illumina HiSeq 2000 sequencer by Lan Lin et al.[34], comprising 14 samples from Brodmann area 4 motor cortex, 7 affected by HD and 7 controls. The dataset, available under the BioProject accession PRJNA316625 at ENA database (https://www.ebi.ac.uk/ena), includes, on average, 152 million raw paired-end reads (100 bases × 2) per sample, whose libraries were prepared using a strand-oriented protocol (Table 1). Known A-to-I RNA editing sites are derived from REDIportal, limiting the analysis to recoding events (i.e., non-synonymous changes induced by RNA editing), because many of them have been linked to a variety of neurodegenerative disorders[35–38]. The procedure begins with the download of RNAseq data from BioProject PRJNA316625 (ref. [34]), and the subsequent preprocessing is carried out as described above to obtain a BAM file of aligned and cleaned reads per sample. Next, `REDItoolDnaRna.py` is launched to interrogate known recoding editing sites stored in REDIportal without any genomic information from WGS/WXS reads. Resulting tables are collected and compared to identify differential A-to-I sites between HD and control groups (Fig. 4), using a simple Python script released in the latest REDItools package.

Both procedures do not require programming expertise. However, they assume familiarity with the Unix/Linux command-line interface, and users should be able to run programs through command line.

Alternative analysis packages

To analyze RNAseq reads, a plethora of tools have been released. This is mainly true for programs devoted to read mapping and quality assessments of sequencing runs. In the preprocessing steps of our procedures, we use FASTQC (https://github.com/s-andrews/FastQC) and FASTP[23] for quality check and adapter removal, while STAR[24] and BWA[25] are suggested to perform genome mapping of transcriptomic and genomic reads, respectively. Nonetheless, other packages can be employed to perform the same steps with similar results. To check the quality of RNAseq reads, for example, alternative packages such as RSeQC[39] or RNA-SeQC[40] can be used instead of FASTQC. Read trimming can also be carried out using programs such as TrimGalore (https://github.com/FelixKrueger/TrimGalore), Trimmomatic[41], Cutadapt[42], fqtrim (https://ccb.jhu.edu/software/fqtrim/) or NGS QC Toolkit[43].

Genome mapping of WGS/WXS and RNAseq reads is the most important preprocessing step because it represents our hypothesis about the genomic origin of reads. Instead of STAR[24], RNAseq

Box 2 | REDItools accessory scripts

The REDItools package comprises auxiliary Python scripts to facilitate the manipulation of output and input tables.

- FilterTable.py: filters in or out positions from a REDItool output table;
  Input: REDItool table and a list of positions to filter in gtf format;
  Output: filtered REDItool table;
- AnnotateTable.py: annotates positions of REDItool output tables;
  Input: REDItool table and indexed annotations in gtf format;
  Output: REDItool table with extra columns including annotations;
- SearchInTable.py: searches positions in REDItool tables;
  Input: REDItool table and a list of positions to look for;
  Output: a list of found positions;
- selectPositions.py: filters out positions from REDItool tables;
  Input: REDItool table and specific parameters to filter sites;
  Output: filtered REDItool table;
- readPsl.py: reads Blat output to detect mis-mapping reads;
  Input: PSL file from Blat or PBlat;
  Output: list of mis-mapping reads for use in REDItools;
- SortTable.py: sorts an input table;
  Input: REDItool table or text-delimited table;
  Output: sorted table;
- SortGFF.py: sorts an input GFF/GTF file;
  Input: GFF/GTF file;
  Output: sorted GFF/GTF file;
- tableToTabix.py: converts an input table to a tabix indexed table;
  Input: REDItool table or text-delimited table;
  Output: tabix indexed table
- GFFToTabix.py: indexes an input GFF/GTF;
  Input: GFF/GTF file;
  Output: tabix indexed GFF/GTF file;
- TableToGFF.py: converts an input REDItool table to GFF/GTF;
  Input: REDItool table;
  Output: GFF/GTF file.
- rediportal2recoding.py: extracts recoding events from REDIportal tables;
  Input: REDIportal table;
  Output: GTF table of recoding events
- subCount.py: counts substitutions in REDItool tables;
  Input: REDItool table;
  Output: observed substitutions in textual format
- get_DE_events.py: detect differential RNA editing;
  Input: REDItool tables and a sample information file;
  Output: list of significant RNA editing events.

reads can be alternatively mapped onto the reference genome using programs like GSNAP[44] or HISAT[45] that are fast and highly accurate in aligning spliced reads. Output results, however, need further steps to be converted in sorted BAM files. SAMtools[26] are quite useful for this purpose. Genomic reads from WGS/WXS experiments can also be mapped using Bowtie2[46] or SOAP2[47].

Regarding the detection of RNA editing events, in the last few years, alternative software to our REDItools package has been released such as RNAEditor[48], RES-Scanner[49], GIREMI[50] and JACUSA[51]. RNAEditor implements a workflow in which reads are mapped by BWA, and RNA variants are filtered according to their genomic location, position within reads and known SNPs, but outputs only variants corresponding to A-to-I candidates[48]. RES-Scanner is a software package to detect and annotate RNA editing sites with matching RNAseq and WGS/WXS data. It invokes BWA to align reads to the reference genome and exonic sequences surrounding known splicing junctions and implements binomial statistics to calculate most likely RNA editing events[49]. GIREMI is another tool to predict RNA editing sites using RNAseq data only and is based on mutual information to discriminate editing events from SNPs[50]. JACUSA is a program to quickly detect RNA editing sites, enabling a set of positional filters and a statistical model for variant calling[51]. The behavior of all these programs in calling RNA editing has been recently comparatively assessed and revised[17]. Unlike these software packages, REDItools implements empirical filters that can be tuned depending on the experimental features[20]. In addition, it can print out all genomic positions covered by RNAseq reads independently from variations. It presents a great advantage when multiple experiments are analyzed because it allows users to know if a given position is supported by RNA reads but not edited across all samples.

Box 3 | REDItools output table

REDItools printout results in simple textual tables. An example is reported below.

Region	Position	Reference	Strand	Coverage-q25	MeanQ	BaseCount[A,C,G,T]	AllSubs	Frequency
chr21	15412990	A	1	18	37.22	[3, 0, 15, 0]	AG	0.83
chr21	15415901	A	1	13	37.15	[2, 0, 11, 0]	AG	0.85
chr21	15423330	A	1	11	38.27	[4, 0, 7, 0]	AG	0.64
chr21	15425640	A	1	8	36.12	[0, 0, 8, 0]	AG	1.00
chr21	15456434	T	1	90	34.96	[0, 6, 1, 83]	TC TG	0.07
chr21	15461406	A	1	83	37.27	[73, 0, 10, 0]	AG	0.12
chr21	15461417	A	1	90	36.26	[72, 0, 18, 0]	AG	0.20
chr21	15461444	A	1	64	37.22	[26, 0, 38, 0]	AG	0.59
chr21	15461479	A	1	70	36.96	[66, 0, 4, 0]	AG	0.06
chr21	15461486	A	1	68	37.06	[61, 0, 7, 0]	AG	0.10
chr21	15461503	A	1	76	37.26	[69, 0, 7, 0]	AG	0.09
chr21	15461511	A	1	81	37.68	[55, 0, 26, 0]	AG	0.32

Each column indicates:
- Region: the genomic region according to reference;
- Position: the exact genomic coordinate (1-based);
- Reference: the nucleotide base at the reference position;
- Strand: strand information with notation 1 for + strand, 0 for - strand and 2 for unknown or non-defined strand;
- Coverage-qxx: the depth per site at a given xx quality score (minimum value);
- MeanQ: the mean quality score per site;
- BaseCount[A,C,G,T]: the base distribution per site in the order A, C, G and T;
- AllSubs: the list of observed substitutions at a given site, separated by a space. A character '-' is included in the case of invariant sites;
- Frequency: the frequency of observed substitution. In the case of multiple substitutions, it refers to the first reported in the AllSubs field.

REDItoolDnaRna.py includes five additional columns to take into account information from WGS/WXS reads. Such columns are indicated as:
- gCoverage-qxx: the depth per site at a given xx quality score (minimum value) in WGS/WXS data;
- gMeanQ: the mean quality score per site in WGS/WXS data;
- gBaseCount[A,C,G,T]: the base distribution per site in the order A, C, G and T;
- gAllSubs: the list of observed substitutions at a given site, separated by a space. A character '-' is included in the case of invariant sites;
- gFrequency: the frequency of observed substitution. In the case of multiple substitutions, it refers to the first reported in the gAllSubs field.

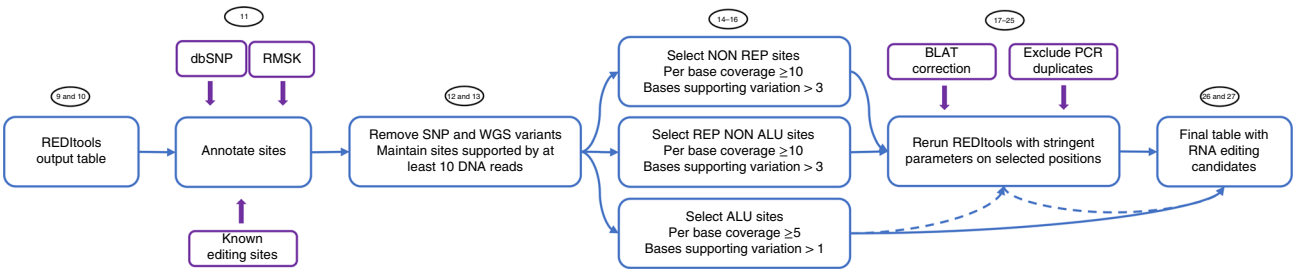

Fig. 3 | Filtering of REDItools tables to call RNA editing events. REDItools tables (Steps 9 and 10, Procedure 1) are generally subjected to further filters to remove potential artifacts and errors. The procedure begins with the annotation of all individual positions using known SNP sites, repeated elements in RepeatMasker and known editing events stored in the REDIportal database (Step 11, Procedure 1). Then, SNPs and sites not supported by ≥10 WGS reads are removed (Steps 12 and 13, Procedure 1) and divided into three groups: ALU, REP NON ALU and NON REP (Steps 14–16, Procedure 1). NON REP and REP NON ALU sites undergo more stringent call criteria than ALU sites that take into account mis-mapping reads and PCR duplicates (Steps 17–25, Procedure 1). Optionally, stringent filters can also be applied to ALU sites (Steps 20 and 21, Procedure 1). Finally, filtered positions are collected in the final list of RNA editing candidates (Steps 26 and 27, Procedure 1).

In addition, other tools to detect RNA editing events based on machine learning approaches have been developed such as RDDpred[52], RED-ML[53] and DeepRed[54]. Although training datasets consisting of lists of known RNA editing events are required, they promise to be quite fast and accurate as large collections of A-to-I sites become available.

Unlike the aforementioned tools, a method to identify hyper-edited regions rescuing RNA reads discarded by mapping software has been recently described[55]. It is particularly useful to detect hyper-editing in short reads in which the rate of unmapped sequences is relatively high. The same approach

Table 1 | RNAseq samples from BioProject PRJNA316625

Sample (run accession)	Status	Reads number
SRR3306823	HD	145.5M
SRR3306824	HD	128.5M
SRR3306825	HD	138.7M
SRR3306826	HD	139.6M
SRR3306827	HD	140.2M
SRR3306828	HD	150.9M
SRR3306829	HD	152.3M
SRR3306830	Control	171.9M
SRR3306831	Control	152.4M
SRR3306832	Control	162.5M
SRR3306833	Control	163.2M
SRR3306834	Control	145.5M
SRR3306835	Control	160.6M
SRR3306836	Control	175M

We report the run accession ID, the diseased status and the number of raw reads per sample (M, million).

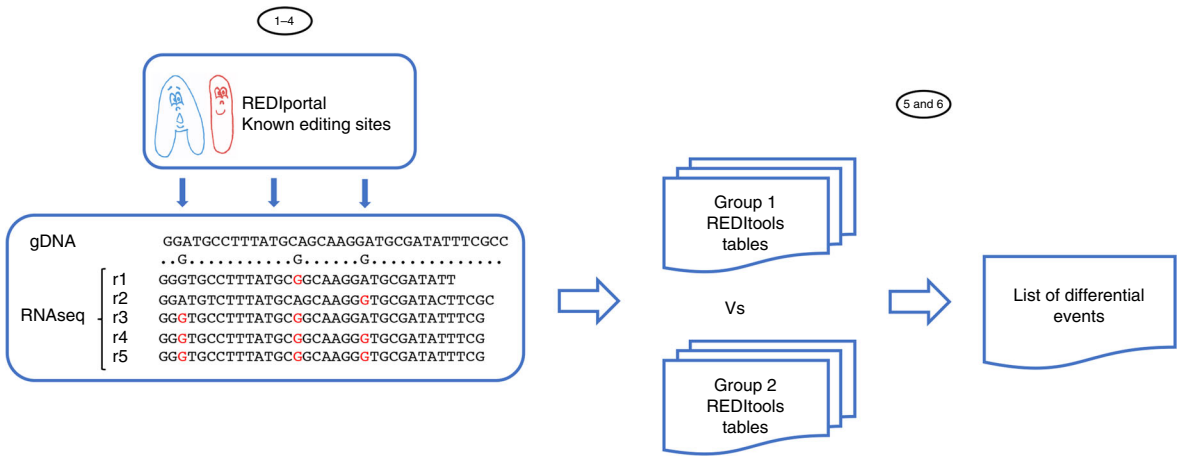

Fig. 4 | Differential RNA editing using REDItools and REDIportal. REDItools and REDIportal can be used in combination to identify differential RNA editing at known sites, i.e., genomic positions in which RNA editing levels are statistically different between two or more conditions. The procedure requires aligned RNAseq data and begins with the selection of known events to explore from REDIportal. Then, REDItools are launched on each RNAseq sample, providing a series of output tables (Steps 1–4, Procedure 2). All REDItools tables are parsed and compared according to metadata, returning a final list of known sites with indications of differential editing (Steps 5 and 6, Procedure 2).

has been implemented in SPRINT[56], a tool able to identify RNA editing events without the need to filter out SNPs.

As an alternative to REDIportal[21], users can use known RNA editing events detected in their own samples or downloaded from other web resources such as DARNED[57] and RADAR[58].

Limitations of the protocol and software

Our procedures have been conceived to work with RNAseq data from a variety of organisms for which a complete genome is available. Since REDItools are based on empirical filters depending on input RNAseq reads, running parameters to detect RNA editing events in users' own RNAseq data should be tuned accordingly.

REDItools have been conceived to be portable and computationally efficient. Nonetheless, the analysis of large datasets could be time consuming because they traverse the reference genome position by position and are sensitive to the coverage and depth of input RNAseq data. To reduce the

execution time, users can run the software in multi-threaded computers or computing farms. In addition, REDItools run on Linux/Unix machines because they are based on the Pysam module, which, in its current release, cannot be compiled on Windows platforms.

The accuracy of RNA editing calls depends on the quality and characteristics of input RNAseq reads, as well as on the parameters used to filter candidates. Paired-end reads should be preferred to single-end reads to mitigate the mismapping effect, and reads of ≥75 bp long should be used. In addition, protocols used for generating RNAseq libraries can also affect the accuracy and yield of final results. Protocols preserving the strand orientation of reads should be preferred because they improve the mapping step and facilitate the correct identification of mismatch type, while protocols capturing total RNA (not limited to the polyA$^+$ fraction) should be used to investigate RNA editing in the non-coding fraction of the transcriptome.

REDItools accept as input pre-aligned reads in BAM format. As a consequence, raw fastq files are not suitable as they are.

For the differential RNA editing analysis, known RNA editing events can be downloaded from our REDIportal[21] database or other existing resources such as DARNED[57] and RADAR[58]. Users are also free to use editing sites obtained by alternative programs. To be used, known editing positions need to be processed as discussed below.

The protocol and software here described can also be applied to input BAM files from long reads generated by third-generation sequencing such as PacBio and ONT (Oxford Nanopore). However, the accuracy of RNA editing calls is not expected to be high because of their larger mismatch error rate and the as-yet-limited throughput of these technologies. Reliable results will probably require suitably developed cleaning procedures and filtering steps.

Applications of the protocol

Since its first release in 2013, REDItools have been successfully used to profile RNA editing in human RNAseq data, demonstrating the tissue specificity of this phenomenon and its pervasive nature in human transcriptomes[10,20]. Through REDItools, A-to-I RNA editing has been investigated in thousands of RNAseq experiments from several projects, and >4.5 million events across 55 human body sites have been collected in the specialized REDIportal database[21]. REDItools and REDIportal have also been used in combination to study the inosinome in single cells, revealing that RNA editing is a cell type–specific mechanism[59]. They have proved useful also in investigating RNA editing in human diseases[35,60–62]. For example, in glioblastoma multiforme, REDItools and REDIportal have revealed that an editing signature can stratify glioblastoma multiforme (GBM) patients and identify gender-dependent high-risk patients[14].

REDItools are not human specific, and, indeed, they have been applied to investigate RNA editing in *Bombus terrestris*[63], *Neurospora crassa*[64], *Symbiodinium microadriaticum*[65], *Drosophila melanogaster*[66] and other non-human organisms. Successful attempts have been carried out also in plant mitochondria[67,68].

Experimental design

To illustrate the potential of REDItools and REDIportal, we describe in detail two procedures that should correspond closely to many users' designs. Both of them require a few common preprocessing steps before the RNA editing calling by REDItools.

Preprocessing of RNAseq data: Steps 1–5 (Procedure 1)

RNAseq data are initially downloaded from the ENA database (https://www.ebi.ac.uk/ena) in the standard fastq format, and the quality of each run is assayed using FASTQC metrics (https://github.com/s-andrews/FastQC). Users should pay attention to the per base sequence content plot showing the distribution of all four bases per each position along the read length. In a random library, the composition of each base is expected to be constant along the read. However, RNAseq libraries produced by priming using random hexamers show biases in the first 10–15 positions that may alter the call of RNA editing variants and, thus, should be excluded in downstream analyses. Contaminants, low quality read regions, adapters and long homopolymeric stretches at the 3′ end of reads are removed using FASTP[23]. Although quality check and trimming are not mandatory steps, they should not be skipped, because they increase the mappability rate and improve the alignment quality, two factors that directly affect the yield and accuracy of final RNA editing results[17].

Cleaned RNAseq reads are mapped onto the reference human genome, assembly hg19, using STAR[24]. Before the alignment, the human genome is downloaded in fasta format from GENCODE

(https://www.gencodegenes.org/) and indexed. To improve the quality of alignments in regions surrounding splice sites, the indices are created using a list of known junctions, automatically extracted by STAR from an annotation file in gff3 format (https://www.ensembl.org/info/website/upload/gff3.html). For the human genome, annotations are downloaded from the GENCODE web site (https://www.gencodegenes.org/). STAR requires ≥30 GB of RAM to complete the mapping but is quite fast and accurate and outputs alignments directly in the standard BAM format, ready for downstream analyses.

RNAseq reads are aligned on the hg19 human genome version because current resources dealing with RNA editing annotations are based on this assembly release.

Preprocessing of WGS data: Steps 6–7 (Procedure 1)

WGS data are extremely important for discovering reliable RNA editing events[15,20]. Indeed, they are currently used to filter out potential SNPs. In the absence of genomic reads, annotations from dbSNP (https://www.ncbi.nlm.nih.gov/snp/) can be used as well. The preprocessing of WGS data is quite similar to the procedure used for RNAseq data and requires the downloading of raw reads from the ENA database, the quality check by FASTQC and the trimming of low-quality regions by FASTP. The mapping is performed using a non-splicing aware software such as BWA[25], after the genome indexing employing the hg19 assembly version. Since this step is time consuming, involving the alignment of >600 million reads, to save time in the downstream analyses, the first procedure is restricted to chromosome 4 only.

Unlike STAR, BWA outputs alignments in SAM format, and, thus, a supplementary step by SAMtools is needed to convert them in a sorted BAM file useful for REDItools.

RNA editing detection in the NA12878 cell line by REDItools using RNAseq and WGS data: Steps 8–27 (Procedure 1)

Inosine is commonly recognized as guanosine by cellular machineries for splicing and translation as well as by sequencing enzymes. For this reason, A-to-I variants appear as A-to-G substitutions (or T-to-C on the opposite strand in the case of non-strand-oriented RNAseq data) in multiple alignments of RNAseq reads. In the NA12878 cell line, A-to-I RNA editing events are detected using our REDItools package[20] (https://github.com/BioinfoUNIBA/REDItools). It comprises three main scripts in the portable Python programming language: (i) REDItoolDnaRNA.py to identify RNA editing candidates by comparing pre-aligned RNAseq and WGS reads in BAM format, (ii) REDItoolKnown.py to explore known RNA editing events in RNAseq experiments, and (iii) REDItoolDenovo.py to detect de novo RNA editing candidates using RNAseq data alone and without any a priori knowledge of genomic information[20,69].

REDItoolDenovo.py implements a very stringent algorithm and is suitable for discovering mammalian recoding events or editing changes in plant mitochondria[67]. REDItoolKnown.py works only on user predefined lists of candidate events, while REDItoolDnaRNA.py is the script used in the majority of situations[20,69,70]. Indeed, it can predict editing changes using RNAseq and WGS reads from the same sample/individual or RNAseq reads alone. It can also work with an input list of known positions in gtf format (https://www.ensembl.org/info/website/upload/gff.html).

REDItoolDnaRNA.py explores genomic positions site by site by applying empirical filters to minimize biases due to sequencing errors, mapping errors and SNPs. It is based on the Pysam module (https://github.com/pysam-developers/pysam), a wrapper of the widely used SAMtools[26], which includes methods and functions to handle read alignments in SAM/BAM format. REDItoolDnaRNA.py has been conceived to handle millions of genomic positions through a multithreaded approach in which input genomic coordinates are split into chunks depending on available cores. For each genomic position covered by RNAseq reads, various filters can be applied such as read coverage, base quality, mapping quality, reads supporting the variation, substitution type and frequency (see Box 1). In addition, positions in homopolymeric regions of predefined length or in intronic sequences surrounding known splice sites can be filtered out, as well as invariant RNAseq positions or sites not supported by WGS. REDItoolDnaRna.py can also exclude duplicated reads or sequences known to align on multiple genome locations by Blat. For stranded RNAseq data, it can infer the strand for each position, thus mitigating biases due to antisense transcription or mapping errors.

At the end of each run, REDItoolDnaRna.py returns a textual table containing several fields such as the coverage depth, the mean quality score, the observed base distribution, the strand

(if available), the list of observed substitutions and the frequency of variation. An example and its description is in Box 3.

For our purposes, REDItoolDnaRNA.py is applied to the transcriptome and genome BAM files from the NA12878 cell line, limiting the analysis to chromosome 4 to save time. Initially the script is launched using non-stringent filters in to recover all genomic positions covered by at least one RNAseq read (Fig. 2). Next, the output table undergoes several filtering steps. In particular, all positions are annotated using dbSNP, RepeatMasker and the full list of known A-to-I events from REDIportal. SNP annotations are used in combination with WGS reads to exclude genomic polymorphisms, while RepeatMasker annotations are employed to assign positions to repetitive elements. REDIportal annotations, however, are included to avoid the prediction at sites already known to be edited. Users working with organisms not included in REDIportal can skip this step. All annotation rounds are performed through the `AnnotateTable.py` script that is part of the REDItools package (accessory scripts are described in Box 2). It works on REDItools output tables proving annotations in gtf format indexed by tabix (a tool included in the htslib software: https://github.com/samtools/htslib).

Annotated positions are then selected to filter out sites not supported by ≥10 WGS reads, SNPs and known A-to-I events. The `selectPositions.py` script from REDItools is used in combination with `awk` to accomplish these steps.

Since RNA editing in humans is pervasive in repetitive elements (especially in Alu) and rare in other genomic regions, selected positions are split and classified in three groups (ALU, REP NON ALU and NON REP) to apply ad hoc filters. ALU sites are all positions located in Alu repetitive elements. For these sites, the RNA editing prediction is quite simple because >97% of known A-to-I events reside in Alu elements[9,10]. For ALU sites, a position is selected as an editing candidate if covered by a minimum of five RNAseq reads and at least one of them supports the variation. In contrast, REP NON ALU sites include positions falling in non-Alu repetitive elements, whereas NON REP sites are located in non-repetitive regions of the genome. For REP NON ALU and NON REP sites, the RNA editing detection is slightly more complicated because only a small fraction of events is located in these genomic regions[71]. Therefore, a position falling in repetitive non-Alu elements or in non-repetitive regions is marked as a potential RNA editing candidate if covered by ≥10 RNAseq reads and a minimum of three bases support the variation. In addition, selected REP NON ALU and NON REP sites undergo further stringent criteria. To exclude false candidates due to mis-mapping errors, reads harboring the variation (mismatch from reference) are aligned onto the human genome using PBLAT[27]. Only reads mapping uniquely (the second-best hit has a score <95% of the best hit) are retained. PCR duplicates are also excluded, collecting all aligned reads supporting REP NON ALU and NON REP sites and applying the SAMtools markdup method. Finally, non-Alu candidates are further subjected to a second run of REDItoolDnaRNA.py, removing sites with variation frequency <0.1 and falling in simple repeats according to RepeatMasker annotation, located within 4 bp of all known splicing junctions according to Gencode gene annotations or in homopolymer runs of ≥5 bp. At this step, REDItoolDnaRNA.py also excludes positions in the first 11 and last 6 bases of reads, to take into account biases due to random priming.

Optionally, sites in Alu elements can also be refined using a second round of REDItoolDnaRNA.py.

Alu elements are primate-specific repeats belonging to the class of retroelements termed short interspersed elements and comprise 11% of the human genome[72]. In case of RNA editing detection in non-primate organisms, the splitting of REDItools positions in ALU, REP NON ALU and NON REP groups can be skipped. In general, however, since A-to-I editing is generally over-represented in double RNA strands[73] formed by repetitive elements in opposite orientation, we strongly suggest dividing annotated positions in at least two groups falling in repetitive or non-repetitive elements and applying different filtering criteria to them.

After the filtering, all remaining sites from Alu and non-Alu regions are collected in the final table of A-to-I candidates, and the distribution of resulting substitutions is calculated.

Differential RNA editing in HD disorder: Steps 1–6 (Procedure 2)

To discover potential A-to-I RNA editing events in HD, we employ known RNA editing events stored in REDIportal (http://srv00.recas.ba.infn.it/atlas/index.html). At the beginning, RNAseq from control and HD samples (BioProject PRJNA316625) are downloaded from the ENA database in fastq format and preprocessed according to the steps described above. Known RNA editing sites are downloaded from REDIportal, and non-synonymous positions are selected and converted in gtf using an accessory Python script (included in the REDItools package). Next, `REDItoolDnaRNA.py` is invoked on

each RNAseq (sorted BAM file), providing the list of editing positions to explore and using non-stringent parameters. Output tables are finally collected, and editing levels per individual positions are compared between control and HD samples to identify sites showing statistically significant differential editing. The comparison is performed by a Python script included in the latest REDItools release. Only sites supported by ≥10 RNAseq reads are taken into account for the differential analysis. Our custom script calculates a Mann–Whitney *P* value for each site if covered in ≥50% of samples per group. Users could also perform a multiple testing correction like Benjamini–Hochberg or Bonferroni to increase the stringency.

Materials

Equipment

- Hardware: 64-bit computer running Linux, Mac OS or other Unix-based operating system with ≥32 GB of RAM (for genome mapping using STAR) and ≥1 TB of free disk space. The software may run on Windows with few modifications, but we have not tested it in a Windows environment.
- The Conda environment manager (version 4.6.14 or later) (https://docs.conda.io/en/latest/) and the following packages from Bioconda channel (https://daler.github.io/bioconda-docs/index.html):
  - bcftools 1.9
  - bedtools 2.28.0
  - bwa 0.7.17
  - bx-python 0.8.2
  - fastp 0.20.0
  - fastqc 0.11.8
  - fisher 0.1.4 (optional)
  - gmap 2018.07.04
  - htslib 1.9
  - libdeflate 1.0
  - pysam 0.15.2 (required by REDItools)
  - rseqc 2.6.4
  - samtools 1.9
  - star 2.7.0f and conda-forge (https://conda-forge.org/):
  - bzip2 1.0.6 (if not already installed on your operating system)
  - git 2.21.0 (if not already installed on your operating system)
  - numpy 1.16.2
  - scipy 1.2.1
  - wget 1.20.1 (if not already installed on your operating system)
- Software not available as Conda package:
  - REDItools 1.3 and accessory scripts (https://github.com/BioinfoUNIBA/REDItools)
  - pblat (blat with multi-threads support) (http://icebert.github.io/pblat/)
  - REDIportal database (http://srv00.recas.ba.infn.it/atlas/index.html)
  - Additional Python scripts developed for this protocol are available at the REDItools repository (https://github.com/BioinfoUNIBA/REDItools/tree/master/NPscripts)

Required data

RNAseq and WGS reads from the NA12878 cell line can be obtained from the ENA database under the accessions SRR1258218 (11 GB) and ERR262997 (325 GB) for RNAseq and WGS, respectively.

RNAseq data from BioProject PRJNA316625 (425 GB) can be downloaded from the ENA database under the accessions reported in Table 1.

In addition, the protocol requires:

- The human reference genome (3 GB) (ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_30/GRCh37_mapping/GRCh37.primary_assembly.genome.fa.gz)
- Related gene annotations (1.4 GB) (ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_30/GRCh37_mapping/gencode.v30lift37.annotation.gtf.gz)
- Known genomic SNPs (86 GB) (http://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/snp151.txt.gz)
- Repetitive elements from RepeatMasker (435 MB) (http://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/rmsk.txt.gz)
- A-to-I sites from REDIportal database (519 MB) (http://srv00.recas.ba.infn.it/webshare/rediportalDownload/table1_full.txt.gz)

Equipment setup

Downloading and installing Conda

Conda and the previously listed packages (Equipment) must be installed to run the complete procedure. Conda is an open source cross-platform package manager and environment control system. It allows users to manage separate environments on the same computing platform with their own software packages and required libraries. The easiest way to obtain Conda is to install Miniconda, a mini version of Anaconda that contains only Conda and its dependencies (<500 MB of disk space required). It is important to note that Conda and its packages can be installed as normal user ($), and, thus, no root privileges (#) are required. See Box 4 for full instructions. For novice users, we provide a Python script, named `conda_pckgs_installer.py`, to automatically install Conda and all required packages. It can be launched using the command:

```
$ python conda_pckgs_installer.py
```

▲ **CRITICAL** The script is interactive, and users need to answer questions that will appear during the installation process.

Downloading and installing non-Conda packages

REDItools and pblat are not available as a Conda package, and their installation requires different steps. See Box 5.

Downloading and organizing required data

See Box 6 for detailed instructions. Users interested in non-human organisms could download genome and related annotations from the UCSC (http://genome.ucsc.edu/) or Ensembl (https://www.ensembl.org/index.html) websites.

Preparing required data

See Box 7 for detailed instructions. For novice users, we provide a Python script, named `download-prepare-data-NP.py`, to automatically download and prepare all required files. It can be launched using the command:

```
$ python download-prepare-data-NP.py rna_editing_protocol <REDItools
folder> <use paths 0/1>
```

where 'rna_editing_protocol' is the main folder in which all data will be downloaded, <REDItools folder> is the absolute path to the REDItools folder and <use paths 0/1> is a flag to use (1) or not (0) a file with absolute paths to requested programs. If <use paths 0/1> is set to 0 and the above software has been correctly installed, paths to requested programs will be automatically checked.

Computing environment, reference genome and annotations need to be set up, downloaded and prepared only the first time; then, they will be useful for any human RNAseq and WGS/WXS experiment.

Procedure

▲ **CRITICAL** Here, we describe all steps for the two RNA editing pipelines: procedure 1, RNA editing detection in the NA12878 cell line, and procedure 2, differential RNA editing in HD samples from BioProject PRJNA316625.

Procedure 1: RNA editing detection in the NA12878 cell line

Preprocessing and alignment of RNAseq reads onto the genome ● Timing ~1 h

1 Quality check NA12878 RNAseq data by FastQC as follows. FastQC reads raw sequence data and runs a set of quality checks useful for finding potential problems or biases.

```
$ conda activate nature_protocol
$ cd rna_editing_protocol/
$ cd RNASeq_SRR1258218
$ mkdir fastqc
```

Box 4 | Installing Conda and associated packages

To install Conda, download its installer for 64-bit computer architectures including Python2 to the location of your choice (e.g., opt/):

```
$ mkdir opt
$ cd opt
$ wget https://repo.anaconda.com/miniconda/Miniconda2-latest-Linux-x86_64.sh
```

Run the following command to start the installation:

```
$ bash Miniconda2-latest-Linux-x86_64.sh
```

During the installation process, review and accept the License Agreement by pressing 'Enter' and select the installation folder. The installer will start copying files and required modules to the location specified in the previous step.

▲ **CRITICAL** At the final step, you are given the option to append the Conda installation folder to the system's PATH variables. By choosing 'yes' here, it will not be necessary to specify the complete path to Conda commands.
Update your system's PATH variables with the command:

```
$ source ~/bash_profile
```

A popular Conda channel for bioinformatics software is Bioconda, which provides multiple software distributions for computational biology. Use the following command to add the Bioconda channel as well as the other channels on which Bioconda depends. It is important to add them in the given order so that the priority is set correctly.

```
$ conda config --add channels defaults
$ conda config --add channels bioconda
$ conda config --add channels conda-forge
```

Bioconda is now enabled, so any packages on the Bioconda channel can be installed into the current Conda environment.
Before starting the installation, a new virtual environment, named nature_protocol, needs to be created, using the following command:

```
$ conda create -n nature_protocol python=2.7 anaconda
```

Press `y` to proceed. This will install the Python version 2.7 and all the associated anaconda packaged libraries at 'anaconda_path/anaconda/envs/envname'.
To switch into the new virtual environment nature_protocol, use:

```
$ conda activate nature_protocol
```

To end a session in the current environment type:

```
$ conda deactivate
```

To install all required packages in the nature_protocol virtual environment, enter the following commands:

```
$ conda install -n nature_protocol bcftools==1.9
$ conda install -n nature_protocol bedtools==2.28.0
$ conda install -n nature_protocol bzip2==1.0.6
$ conda install -n nature_protocol bwa==0.7.17
$ conda install -n nature_protocol bx-python==0.8.2
$ conda install -n nature_protocol fastp==0.20.0
$ conda install -n nature_protocol fastqc==0.11.8
$ conda install -n nature_protocol fisher==0.1.4 (optional)
$ conda install -n nature_protocol git==2.21.0
$ conda install -n nature_protocol gmap==2018.07.04
$ conda install -n nature_protocol htslib==1.9
$ conda install -n nature_protocol libdeflate==1.0
$ conda install -n nature_protocol numpy==1.16.2
$ conda install -n nature_protocol pysam==0.15.2
$ conda install -n nature_protocol rseqc==2.6.4
$ conda install -n nature_protocol samtools==1.9
$ conda install -n nature_protocol scipy==1.2.1
$ conda install -n nature_protocol star==2.7.0f
$ conda install -n nature_protocol wget==1.20.1
```

Box 5 | Downloading and installing REDItools and pblat

To complete the protocol, we suggest that users create a single directory in which all data and intermediate files can be stored.

▲ **CRITICAL** All commands are described under the assumption that the user is working in this directory.
To create your main working directory, here named rna_editing_protocol, use the command:

```
$ mkdir rna_editing_protocol
```

REDItools require Python 2.7 and the external pysam (latest available version (≥0.15.2) is strongly recommended) module available as a Conda package. Optionally, users interested in running the REDItoolsDeNovo.py script can install the fisher module (≥0.1.4). It is not required for the procedure described here.
To install REDItools, clone its GitHub repository in the main working directory using the commands:

```
$ cd rna_editing_protocol
$ git clone https://github.com/BioinfoUNIBA/REDItools.git
```

Once the download is complete, enter the folder and run the following command to check if the pysam module is already installed:

```
$ cd REDItools
$ python -c 'import pysam'
```

If no errors are returned, the module is on your machine but could be outdated. To check its version, type the command:

```
$ conda list | grep pysam
```

If pysam version is not ≥0.15.2, type the additional command:

```
$ conda install pysam==0.15.2
```

REDItools are stand-alone scripts, and each one can be simply launched by entering the folder in which it is contained:

```
$ cd main
$ python REDItoolDnaRna.py -h
```

Pblat is the parallelized version with multithread support of the BLAT, used to identify mis-mapping reads. Pblat options are available at the UCSC web page (https://genome.ucsc.edu/goldenPath/help/blatSpec.html#blatUsage), and its compilation on Linux and Mac OS machines does not require specific dependencies.
To install pblat, use the commands:

```
$ cd rna_editing_protocol
$ git clone https://github.com/icebert/pblat.git
```

Once the download is complete, to compile the source code, simply enter the source code directory and invoke the 'make' command:

```
$ cd pblat/
$ make
$ cd..
```

```
$ cd fastqc
$ fastqc../SRR1258218_1.fastq.gz../SRR1258218_2.fastq.gz
$ cd..
```

2 Trim NA12878 RNASeq data by fastp:

```
$ mkdir fastp
$ cd fastp
$  fastp  -i../SRR1258218_1.fastq.gz  -I../SRR1258218_2.fastq.gz  -o
out_SRR1258218_1.fastq.gz -O out_SRR1258218_2.fastq.gz -q 25 -u 10 -l
50 -y -x -w 4
```

where -i [read1 input file name], -I [read1 input file name], -o [read1 output file name], -O [read2 output file name], -q [INT] 'base phred quality' (e.g., 25), -u [INT] 'how many percents of bases are allowed to be unqualified' (e.g., 10, means 10%), -l [INT] 'length required, reads shorter than length will be discarded' (e.g., 50), -y 'low complexity filter, the complexity is defined as the percentage of

Box 6 | Downloading and organizing required data

Enter into the main working directory

```
$ cd rna_editing_protocol/
```

Download and unzip the human reference genome hg19 in FASTA format ● Timing ~1 min:

```
$ mkdir genome_hg19
$ cd genome_hg19/
$ wget ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_30/GRCh37_mapping/GRCh37.
primary_assembly.genome.fa.gz
$ gunzip GRCh37.primary_assembly.genome.fa.gz
$ cd..
```

Download and unzip Gencode annotations in GTF format ● Timing ~30 s

```
$ mkdir Gencode_annotation
$ cd Gencode_annotation
$ wget ftp://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_30/GRCh37_mapping/gencode.
v30lift37.annotation.gtf.gz
$ gunzip gencode.v30lift37.annotation.gtf.gz
$ cd..
```

Download and unzip hg19 RefSeq annotations in bed format for strand detection ● Timing ~30 s

```
$ mkdir Strand_detection
$ cd Strand_detection
$ wget --no-check-certificate
```

https://sourceforge.net/projects/rseqc/files/BED/Human_Homo_sapiens/hg19_RefSeq.bed.gz

```
$ gunzip hg19_RefSeq.bed.gz
$ cd..
```

Download and unzip RepeatMasker annotations ● Timing ~1 min

```
$ mkdir rmsk
$ cd rmsk
$ wget http://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/rmsk.txt.gz
$ gunzip rmsk.txt.gz
$ cd..
```

Download and unzip dbSNP annotations ● Timing ~23 min

```
$ mkdir snp151
$ cd snp151
$ wget http://hgdownload.soe.ucsc.edu/goldenPath/hg19/database/snp151.txt.gz
$ gunzip snp151.txt.gz
$ cd..
```

Download and unzip REDIportal annotations ● Timing ~1 min

```
$ mkdir rediportal
$ cd rediportal
$ wget http://srv00.recas.ba.infn.it/webshare/rediportalDownload/table1_full.txt.gz
$ gunzip table1_full.txt.gz
$ cd..
```

Download and unzip NA12878 WGS reads ● Timing ~2 h 30 min

```
$ mkdir WGS_ERR262997
$ cd WGS_ERR262997/
$ wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR262/ERR262997/ERR262997_1.fastq.gz
$ wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR262/ERR262997/ERR262997_2.fastq.gz
$ gunzip ERR262997_1.fastq.gz
$ gunzip ERR262997_2.fastq.gz
$ cd..
```

Download NA12878 RNAseq reads ● Timing ~10 min

```
$ mkdir RNASeq_SRR1258218
$ cd RNASeq_SRR1258218/
$wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR125/008/SRR1258218/SRR1258218_1.fastq.gz
$ wget ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR125/008/SRR1258218/SRR1258218_2.fastq.gz
```

Box 6 | Downloading and organizing required data (continued)

Download and unzip RNAseq reads from BioProject PRJNA316625. This step may take several hours depending on your network speed. As an alternative, users may use Aspera Connect (https://downloads.asperasoft.com/connect2/) instead of wget. ● Timing ~30 h

```
$ mkdir PRJNA_316625
$ cd PRJNA_316625
$ mkdir SRR3306823
$ cd SRR3306823
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/003/SRR3306823/SRR3306823_1.fastq.gz
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/003/SRR3306823/SRR3306823_2.fastq.gz
$ cd..
$ mkdir SRR3306824
$ cd SRR3306824
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/004/SRR3306824/SRR3306824_1.fastq.gz
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/004/SRR3306824/SRR3306824_2.fastq.gz
$ cd..
$ mkdir SRR3306825
$ cd SRR3306825
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/005/SRR3306825/SRR3306825_1.fastq.gz
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/005/SRR3306825/SRR3306825_2.fastq.gz
$ cd..
$ mkdir SRR3306826
$ cd SRR3306826
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/006/SRR3306826/SRR3306826_1.fastq.gz
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/006/SRR3306826/SRR3306826_2.fastq.gz
$ cd..
$ mkdir SRR3306827
$ cd SRR3306827
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/007/SRR3306827/SRR3306827_1.fastq.gz
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/007/SRR3306827/SRR3306827_2.fastq.gz
$ cd..
$ mkdir SRR3306828
$ cd SRR3306828
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/008/SRR3306828/SRR3306828_1.fastq.gz
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/008/SRR3306828/SRR3306828_2.fastq.gz
$ cd..
$ mkdir SRR3306829
$ cd SRR3306829
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/009/SRR3306829/SRR3306829_1.fastq.gz
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/009/SRR3306829/SRR3306829_2.fastq.gz
$ cd..
$ mkdir SRR3306830
$ cd SRR3306830
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/000/SRR3306830/SRR3306830_1.fastq.gz
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/000/SRR3306830/SRR3306830_2.fastq.gz
$ cd..
$ mkdir SRR3306831
$ cd SRR3306831
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/001/SRR3306831/SRR3306831_1.fastq.gz
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/001/SRR3306831/SRR3306831_2.fastq.gz
$ cd..
$ mkdir SRR3306832
$ cd SRR3306832
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/002/SRR3306832/SRR3306832_1.fastq.gz
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/002/SRR3306832/SRR3306832_2.fastq.gz
$ cd..
$ mkdir SRR3306833
$ cd SRR3306833
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/003/SRR3306833/SRR3306833_1.fastq.gz
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/003/SRR3306833/SRR3306833_2.fastq.gz
$ cd..
$ mkdir SRR3306834
$ cd SRR3306834
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/004/SRR3306834/SRR3306834_1.fastq.gz
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/004/SRR3306834/SRR3306834_2.fastq.gz
$ cd..
$ mkdir SRR3306835
$ cd SRR3306835
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/005/SRR3306835/SRR3306835_1.fastq.gz
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/005/SRR3306835/SRR3306835_2.fastq.gz
$ cd..
```

Box 6 | Downloading and organizing required data (continued)

```
$ mkdir SRR3306836
$ cd SRR3306836
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/006/SRR3306836/SRR3306836_1.fastq.gz
$ wget ftp.sra.ebi.ac.uk/vol1/fastq/SRR330/006/SRR3306836/SRR3306836_2.fastq.gz
$ cd../../
```

base that is different from its next base (base[i]!= base[i+1])', -x 'enable polyX trimming in 3' ends' and -w 'worker thread number'.

```
$ cd..
```

3 Align NA12878 RNASeq reads to the reference genome with STAR:

```
$ mkdir Alignment
$ cd Alignment/
$ STAR --runThreadN 4 --genomeDir../STAR/STAR_genome_index_ucsc
--genomeLoad NoSharedMemory --outFileNamePrefix SRR1258218_ --outReads
Unmapped Fastx --outSAMtype BAM SortedByCoordinate --outSAMstrand-
Field intronMotif --outSAMattributes All --readFilesCommand zcat --
outFilterType BySJout --outFilterMultimapNmax 1 --alignSJoverhangMin
8 --alignSJDBoverhangMin 1 --outFilterMismatchNmax 999 --outFilter-
MismatchNoverLmax 0.04 --alignIntronMin 20 --alignIntronMax 1000000 --
alignMatesGapMax 1000000 --readFilesIn../RNASeq_SRR1258218/fastp/
out_SRR1258218_1.fastq.gz../RNASeq_SRR1258218/fastq/out_SRR1258218_2.
fastq.gz
```

where:
--runThreadN [INT] is the number of threads on the user's machine (e.g., 4);
--genomeDir is the folder where the indexed genome files are stored;
--genomeLoad NoSharedMemory indicates the program to store a separate copy of reference in the computer's RAM for each alignment job;
--outFileNamePrefix is a string (e.g., SRR1258218_) to be included as the output filename;
--outReadsUnmapped indicates the program to store unmapped and partially mapped reads (mate1/2) in separate fasta/fastq files;
--outSAMtype SortedByCoordinate specifies the alignment output format (e.g., BAM) that will be sorted according to the specified criterion (e.g., coordinates);
--outSAMstrandField intronmotif tells the program to discard reads with inconsistent and/or non-canonical introns;
--outSAMattributes describes which SAM flags will be present in the output (e.g., ALL means NH HI AS nM NM MD jM j);
--readFilesCommand specifies a command line that will be executed for each of the input files (e.g., zcat since the input reads are compressed with gzip);
--outFilterType indicates the program to keep only those reads that contain junctions that passed filtering into SJ.out.tab;
--outFilterMultimapNmax is the maximum number of loci a read is allowed to map to (if set to 1, only unique reads are shown in the output);
--alignSJoverhangMin is the minimum spliced reads overhang (e.g., eight nucleotides);
--alignSJDBoverhangMin is the minimum spliced reads overhang for annotated splicing sites (splice junctions db inferred by STAR from reference genome and annotations);
--outFilterMismatchNmax is the maximum number of mismatches per reads pair (large numbers for this parameter (e.g., 999) switch off the filter);
--outFilterMismatchNoverLmax is the maximum number of tolerated mismatches relative to read length (e.g., 0.04; for 2× 100 b, the maximum number of mismatches is 0.04 × 200 = 8 b for the paired reads);
--alignIntronMin indicates to STAR the minimum intron length to consider during the alignment process (e.g., 20);

Box 7 | Preparing required data

Build a reference genome index for BWA ● Timing ~1 h 30 min

```
cd genome_hg19/
$ bwa index GRCh37.primary_assembly.genome.fa
$ cd..
```

Build a reference genome index for STAR ● Timing ~6 h 15 min

```
$ mkdir STAR
$ cd STAR
$ mkdir STAR_genome_index_ucsc
$ STAR --runMode genomeGenerate --genomeDir STAR_genome_index_ucsc --genomeFastaFiles../genome_hg19/
GRCh37.primary_assembly.genome.fa --sjdbGTFfile../Gencode_annotation/gencode.v30lift37.annotation.gtf
--sjdbOverhang 75
```

where --runMode is the type of run ('genomeGenerate' specifies the genome indices generation job), --genomeDir is the path to the reference genome to be indexed, --sjdbGTFfile indicates the location of the file with annotated transcripts in the standard GTF format (STAR will extract splice junctions from this file and use them to improve the mapping accuracy) and --sjdbOverhang is the length of the genomic sequence surrounding the annotated junction to be used in constructing the splice junctions database.

```
$ cd../../
```

Prepare RepeatMasker annotations for REDItools ● Timing ~2 min

```
$ cd rmsk
$ awk 'OFS="\t"{print $6,
"rmsk_hg19",$12,$7+1,$8,".",$10,".","gene_id \""$11"\"; transcript_id \""$13"\";"}' rmsk.txt > rmsk.gtf
$ sort -k1,1 -k4,4n rmsk.gtf > rmsk.sorted.gtf
$ bgzip rmsk.sorted.gtf
$ tabix -p gff rmsk.sorted.gtf.gz
$ cd..
```

Prepare dbSNP annotations for REDItools ● Timing ~2 h 30 min

```
$ cd snp151
$ awk 'OFS="\t"{if ($11=="genomic" && $12=="single") print $2,"ucsc_snp151_hg19","snp",$4,$4,".",
$7,".","gene_id \""$5"\"; transcript_id \"
"$5"\";"}' snp151.txt > snp151.gtf
$ sort -k1,1 -k4,4n snp151.gtf > snp151.sorted.gtf
$ bgzip snp151.sorted.gtf
$ tabix -p gff snp151.sorted.gtf.gz
$ cd..
```

Prepare splice sites annotations for REDItools ● Timing ~3 min

```
$ cd Gencode_annotation
$ gtf_splicesites gencode.v30lift37.annotation.gtf > splicesites
$ awk -F" " '{split($2,a,":"); split(a[2],b,"."); if (b[1]>b[3]) print a[1],b[3],b[1],toupper(substr
($3,1,1)),"-"; else print a[1],b[1],b[3],toupper(substr($3,1,1)),"+"}' splicesites > gencode.v30lift37.
splicesites.txt
$ cd..
```

Prepare REDIportal annotations for REDItools and extract recoding events ● Timing ~1 min

```
$ cd rediportal
$ awk 'OFS="\t"{sum+=1; print $1,"rediportal","ed",$2,$2,".",$5,".","gene_id \""sum"\"; transcript_id
\""sum"\";"}' table1_full.txt > atlas.gtf
$ bgzip atlas.gtf
$ tabix -p gff atlas.gtf.gz
$../REDItools/accessory/rediportal2recoding.py
table1_full.txt > atlas_recoding.gff
$ sort -V -k1,1 -k4,4n atlas_recoding.gff > srtd_atlas_recoding.gff
$ bgzip srtd_atlas_recoding.gff
$ tabix -p gff srtd_atlas_recoding.gff
$ cd..
```

Create the `nochr` file for REDItools ● Timing ~20 s

```
$ cd genome_hg19/
$ grep ">" GRCh37.primary_assembly.genome.fa | awk '{if (substr($1,1,3)==">GL") print $2}' > nochr
$ cd..
```

Box 7 | Preparing required data (continued)

Index the reference genome for REDItools ● Timing ~30 s

```
$ cd genome_hg19/
$ samtools faidx GRCh37.primary_assembly.genome.fa
```

--alignIntronMax is the maximum intron length (e.g., 1,000,000);
--alignMatesGapMax is the maximum genomic distance between mates (e.g., 1,000,000);
--readFilesIn is the path to input read1 (and, if present, read2).

▲ **CRITICAL STEP** STAR alignment can be split between multiple threads with significant changes in mapping speed. The number of threads that is selected with --runThreadN is suggested to be less than or equal to the number of physical processors (cores) of the computer. If other processes are running, this number may need to be reduced. In systems with support for hyper-threading, the mapping speed can be further increased by setting Nthread up to twice the number of physical cores.

▲ **CRITICAL STEP** STAR is quite fast but requires ≥30 GB of RAM memory for the human genome. Users can calculate the memory requirement according to the following empirical formula: 10 × genome size in GB (for ~3 GB of human genome, STAR will require ~30 GB of RAM).

4 Index aligned NA12878 RNAseq reads with SAMtools:

```
$ samtools index SRR1258218_Aligned.sortedByCoord.out.bam
$ cd..
```

Detection of the strand orientation of RNAseq reads ● Timing ~1 min

5 Infer the orientation of NA12878 RNASeq reads using the infer_experiment.py script from the RSeQC package. It compares the orientation of reads with the orientation of known transcript annotations. For further details, please refer to RSeQC homepage at http://rseqc.sourceforge.net/.

```
$ cd Strand_detection/
$ infer_experiment.py -r hg19_RefSeq.bed -s 2000000 -i../Alignment/
SRR1258218_Aligned.sortedByCoord.out.bam
```

where:
-r [REFGENE_BED] is the gene annotations file (in BED format) for the reference human genome assembly hg19;
-s [SAMPLE_SIZE] is the number of reads sampled from SAM/BAM to infer strand orientation;
-i [INPUT_FILE] is the input alignment file in SAM or BAM format.

```
$ cd..
```

Alignment of WGS reads onto the genome ● Timing ~30 h

6 Align NA12878 WGS reads to the reference genome with BWA mem:

```
$ cd WGS_ERR262997/
$ bwa mem -t 24../genome_hg19/GRCh37.primary_assembly.genome.fa
-Y ERR262997_1.fastq ERR262997_2.fastq > ERR262997.sam
```

where -t [INT] is the 'Number of threads' on the user's machine, and -Y indicates the 'Use soft clipping for supplementary alignments'.

7 Select genomic reads mapping on chromosome 4 only:

```
$ mkdir chr4_reads
$ cd chr4_reads/
$  awk  '/^chr4\t/  {printf("%s\t0\t%s\n",$1,$2);}'../../genome_hg19/
GRCh37.primary_assembly.genome.fa.fai > chr4.bed
$ samtools view -b -F4 -L chr4.bed -o chr4.bam -@ 10../ERR262997.sam
```

where -L [FILE] 'only include reads overlapping a BED FILE' (e.g., chr4.bed), -b [output BAM format], -o FILE [output file name] (e.g., chr4.bam), -F INT 'only include reads with none of the FLAGS in INT present' (e.g., F4 to exclude unmapped reads) and -@ [INT] 'Number of additional threads to use'.

```
$ samtools sort chr4.bam > sorted_chr4.bam
$ samtools index sorted_chr4.bam
$ cd../../
```

Identification of all DNA–RNA variants ● Timing ~3 h

8 Detect all potential DNA–RNA variants in the NA12878 cell line (limited to chromosome 4) using the REDItoolDnaRNA.py script:

```
$ mkdir Reditool_DNA_RNA_chr4
$ cd Reditool_DNA_RNA_chr4
$../REDItools/main/REDItoolDnaRna.py     -i../Alignment/SRR1258218_
Aligned.sortedByCoord.out.bam -j../WGS_ERR262997/chr4_reads/sorted_
chr4.bam -o editing_chr4_1_191154276 -f../genome_hg19/GRCh37.primary_
assembly.genome.fa -t10 -c1,1 -m30,255 -v1 -q30,30 -e -n0.0 -N0.0 -u -l
-p -s2 -g2 -S -Y chr4:1-191154276
```

where:
-i refers to RNAseq in BAM format;
-j refers to DNaseq BAM file;
-o is the output folder [rediFolder_xxxx by default] in which all results will be stored; xxxx is a random number used to distinguish between different runs;
-t is the number of threads for the job (e.g., 10). It indicates how many processes should be launched and assigned to different threads. Each thread will work on an individual chromosome/region;
-m is the minimum mapping quality score for WGS and RNAseq reads, respectively (e.g., 30,255); it should be tuned depending on the alignment program.
-v is the minimum number of RNAseq reads supporting the variation (e.g., 1);
-q is the minimum quality score for WGS and RNAseq reads, respectively (e.g., 30,30);
-e indicates the script to exclude multiple hits in RNAseq;
-n is the minimum RNA editing frequency to be considered (e.g., 0.0);
-N is the minimum variation frequency for WGS (e.g., 0.0);
-u tells the script to consider mapping quality set with option -m;
-l indicates to the script to remove substitutions in homopolymeric regions;
-p tells the script to use paired and concordant RNAseq reads only;
-s tells the script to infer strand in the case of strand-oriented reads. Available values are: 1 (read1 as RNA, read2 not as RNA), 2 (read1 not as RNA, read2 as RNA), 12 (read1 as RNA, read2 as RNA) and 0 (read1 not as RNA, read2 not as RNA).
-g indicates the strand inference type: 1 (maxValue) or 2 (useConfidence);
-S is the strand correction. Once the strand has been inferred, only bases according to this strand will be selected;
-Y indicates chromosome coordinates on which the analysis has to be performed.

? TROUBLESHOOTING

▲ **CRITICAL STEP** The option -s2 is mandatory to infer the strand. Its value has been deduced by infer_experiment.py at Step 7. The options -g 2 and -S require the option -s and enable the strand correction. Further details can be found at the REDItools GitHub page (https://github.com/BioinfoUNIBA/REDItools/blob/master/README.md).

▲ **CRITICAL STEP** Input BAM files must be created using the same reference genome file.

Every time an instance of REDItoolDnaRna.py is launched, it creates an output folder and an output table whose names contain a random number as suffix. A typical REDItoolDnaRna.py output folder is rediFolder_xxxx, where xxxx is a random number generated at script run. It contains a second folder, DnaRna_xxxx, which in turn includes the output table outTable_xxxx and the parameters.txt file summarizing the options and values used. Users can modify the prefix name

of the main output folder using the -o option, while the name of the internal folder can be partially changed using the -F option.

Filtering of DNA–RNA variants ● Timing ~30 h

▲ **CRITICAL** The following Steps 9–27 are mandatory to filter out potential errors and improve final results.

9 Enter the REDItoolDnaRna.py output folder:

```
$ cd editing_chr4_1_191154276/
$ cd DnaRna_39649917/
```

10 Exclude invariant positions as well as positions not supported by ≥10 WGS reads:

```
$ awk 'FS="\t" {if ($8!="-" && $10>=10 && $13=="-") print}' outTable_39649917 > outTable_39649917_chr4.out
```

where `$8!="-"` requires only variant positions (from column 8 of the output table), `$10>=10` selects sites covered by ≥10 WGS reads and `$13=="-"` considers only WGS homozygous positions.

11 Annotate positions using RepeatMasker and dbSNP annotations:

```
$ python../../../REDItools/accessory/AnnotateTable.py -a../../../rmsk/rmsk.sorted.gtf.gz -n rmsk -i outTable_39649917_chr4.out -o outTable_39649917_chr4.out.rmsk -u
$ python../../REDItools/Accessory/AnnotateTable.py -a../../../snp151/snp151.sorted.gtf.gz -n snp151 -i outTable_39649917_chr4.out.rmsk -o outTable_39649917_chr4.out.rmsk.snp -u
```

? TROUBLESHOOTING

12 Create a first set of positions selecting sites supported by at least five RNAseq reads and a single mismatch:

```
$ python../../../REDItools/accessory/selectPositions.py -i outTable_39649917_chr4.out.rmsk.snp -c 5 -v 1 -f 0.0 -o outTable_39649917_chr4.out.rmsk.snp.sel1
```

13 Create a second set of positions selecting sites supported by ≥10 RNAseq reads, three mismatches and minimum editing frequency of 0.1:

```
$ python../../../REDItools/accessory/selectPositions.py -i outTable_39649917_chr4.out.rmsk.snp -c 10 -v 3 -f 0.1 -o outTable_39649917_chr4.out.rmsk.snp.sel2
```

14 Select ALU sites from the first set of positions:

```
$ awk 'FS="\t" {if ($1!="chrM" && substr($16,1,3)=="Alu" && $17=="-" && $8!="-" && $10>=10 && $13=="-") print}' outTable_39649917_chr4.out.rmsk.snp.sel1 > outTable_39649917_chr4.out.rmsk.snp.alu
```

where `$1!="chrM"` excludes mitochondrial reads, `substr($16,1,3)=="Alu"` selects positions in Alu elements, `$17=="-"` removes known SNPs, `$8!="-"` requires only variant positions, `$10>=10` selects sites covered by ≥10 WGS reads and `$13=="-"` considers only WGS homozygous positions.

15 Select REP NON ALU sites from the second set of positions, excluding sites in Simple repeats or Low complexity regions:

```
$ awk 'FS="\t" {if ($1!="chrM" && substr($16,1,3)!="Alu" && $15!="-" && $15!="Simple_repeat" && $15!="Low_complexity" && $17=="-" && $8!="-"
```

```
&& $10>=10 && $14<=0.05 && $9>=0.1) print}' outTable_39649917_chr4.
out.rmsk.snp.sel2 > outTable_39649917_chr4.out.rmsk.snp.nonalu
```

where `$1!="chrM"` excludes mitochondrial reads, `substr($16,1,3)!="Alu"` selects positions in non-Alu elements, `$15!="-"` excludes non-annotated sites, `$15!="Simple_repeat"` excludes positions in simple repeats, `$15!="Low_complexity"` excludes sites in low-complexity regions, `$17=="-"` removes known SNPs, `$8!="-"` requires only variant positions, `$10>=10` selects sites covered by ≥10 WGS reads, `$14<=0.05` considers WGS homozygous positions (with the minor allele frequency <0.05) and `$9>=0.1` includes only sites with an editing level ≥0.1.

16 Select NON REP sites from the second set of positions:

```
$ awk 'FS="\t" {if ($1!="chrM" && substr($16,1,3)!="Alu" && $15=="-"
&& $17=="-" && $8!="-" && $10>=10 && $14<=0.05 && $9>=0.1) print}'
outTable_39649917_chr4.out.rmsk.snp.sel2 > outTable_39649917_chr4.
out.rmsk.snp.nonrep
```

where `$1!="chrM"` excludes mitochondrial reads, `substr($16,1,3)!="Alu"` selects positions in non-Alu elements, `$15=="-"` includes sites in non-repetitive elements, `$17=="-"` removes known SNPs, `$8!="-"` requires only variant positions, `$10>=10` selects sites covered by ≥10 WGS reads, `$14<=0.05` considers WGS homozygous positions (with the minor allele frequency <0.05) and `$9>=0.1` includes only sites with an editing level ≥0.1.

17 Annotate ALU, REP NON ALU and NON REP sites using known editing events from REDIportal:

```
$  python../../../REDItools/accessory/AnnotateTable.py  -a../../../
rediportal/atlas.gtf.gz -n ed -k R -c 1 -i outTable_39649917_chr4.out.
rmsk.snp.alu -o outTable_39649917_chr4.out.rmsk.snp.alu.ed -u
$  python../../../REDItools/accessory/AnnotateTable.py  -a../../../
rediportal/atlas.gtf.gz -n ed -k R -c 1 -i outTable_39649917_chr4.out.
rmsk.snp.nonalu -o outTable_39649917_chr4.out.rmsk.snp.nonalu.ed -u
$  python../../../REDItools/accessory/AnnotateTable.py  -a../../../
rediportal/atlas.gtf.gz -n ed -k R -c 1 -i outTable_39649917_chr4.out.
rmsk.snp.nonrep -o outTable_39649917_chr4.out.rmsk.snp.nonrep.ed -u
```

18 Extract known editing events from ALU, REP NON ALU and NON REP sites:

```
$ mv outTable_39649917_chr4.out.rmsk.snp.alu.ed alu
$ mv outTable_39649917_chr4.out.rmsk.snp.nonalu.ed nonalu
$ mv outTable_39649917_chr4.out.rmsk.snp.nonrep.ed nonrep
$ cat alu nonalu nonrep > alu-nonalu-nonrep
$ awk 'FS="\t" {if ($19=="ed") print}' alu-nonalu-nonrep > knownEditing
```

where `$19=="ed"` selects only known RNA editing events.

19 Convert editing candidates in REP NON ALU and NON REP sites in GFF format for further filtering:

```
$ cat nonalu nonrep > nonalu-nonrep
awk 'FS="\t" {if ($19!="ed") print}' nonalu-nonrep > pos.txt
```

where `$19!="ed"` selects novel RNA editing events.

```
$ python../../../REDItools/accessory/TableToGFF.py -i pos.txt -s -t -o
pos.gff
```

20 Convert editing candidates in ALU sites in GFF format for further filtering:

```
$ awk 'FS="\t" {if ($19!="ed") print}' alu > posalu.txt
```

where `$19!="ed"` selects novel RNA editing events.

```
$ python../../../REDItools/accessory/TableToGFF.py -i posalu.txt -s -t
-o posalu.gff
```

21 Launch REDItoolDnaRna.py on ALU sites using stringent criteria to recover potential editing candidates:

```
$ python../../../REDItools/main/REDItoolDnaRna.py -s 2 -g 2 -S -t 4 -i../
../../Alignment/SRR1258218_Aligned.sortedByCoord.out.bam -f../../../
genome_hg19/GRCh37.primary_assembly.genome.fa -c 5,5 -q 30,30 -m
255,255 -O 5,5 -p -u -a 11-6 -l -v 1 -n 0.0 -e -T posalu.sorted.gff.gz -w../
../../Gencode_annotation/ gencode.v30lift37.splicesites.txt -k../
../../ genome_hg19/nochr -R -o firstalu
```

22 Launch REDItoolDnaRna.py on REP NON ALU and NON REP sites using stringent criteria to recover RNAseq reads harboring reference mismatches:

```
$ python../../../REDItools/main/REDItoolDnaRna.py -s 2 -g 2 -S -t 4 -i../
../../Alignment/SRR1258218_Aligned.sortedByCoord.out.bam -f../../../
genome_hg19/GRCh37.primary_assembly.genome.fa -c 10,10 -q 30,30 -m
255,255 -O 5,5 -p -u -a 11-6 -l -v 3 -n 0.1 -e -T pos.sorted.gff.gz -w../../
../Gencode_annotation/ gencode.v30lift37.splicesites.txt -k../../../
genome_hg19/nochr --reads -R --addP -o first
```

23 Launch pblat on RNAseq reads harboring reference mismatches from Step 22 and select multi-mapping reads:

```
$../../../pblat/pblat -t=dna -q=rna -stepSize=5 -repMatch=2253 -min-
Score=20 -minIdentity=0../../../genome_hg19/GRCh37.primary_assembly.
genome.fa first/DnaRna_51144481/outReads_51144481 reads.psl
$ python../../../REDItools/accessory/readPsl.py reads.psl badreads.txt
```

? TROUBLESHOOTING

24 Extract RNAseq reads harboring reference mismatches from Step 22 and remove duplicates:

```
$ sort -k1,1 -k2,2n -k3,3n first/DnaRna_51144481/outPosReads_51144481 |
mergeBed > bed
$ samtools view -@ 4 -L bed -h -b../../../ Alignment/SRR1258218_Aligned.
sortedByCoord.out.bam > SRR1258218_bed.bam
$ samtools sort -@ 4 -n SRR1258218_bed.bam -o SRR1258218_bed_ns.bam
$ samtools fixmate -@ 4 -m SRR1258218_bed_ns.bam SRR1258218_bed_ns_fx.bam
$ samtools sort -@ 4 SRR1258218_bed_ns_fx.bam -o SRR1258218_bed_ns_
fx_st.bam
$ samtools markdup -r -@ 4 SRR1258218_bed_ns_fx_st.bam SRR1258218_bed_
dedup.bam
$ samtools index SRR1258218_bed_dedup.bam
```

? TROUBLESHOOTING

25 Re-run REDItoolDnaRna.py on REP NON ALU and NON REP sites using stringent criteria, deduplicated reads and mis-mapping info:

```
$ python../../../REDItools/main/REDItoolDnaRna.py -s 2 -g 2 -S -t 4 -i
SRR1258218_bed_dedup.bam -f../../../genome_hg19/GRCh37.primary_
assembly.genome.fa -c 10,10 -q 30,30 -m 255,255 -O 5,5 -p -u -a 11-6 -l
-v 3 -n 0.1 -e -T pos.sorted.gff.gz -w../../../Gencode_annotation/
gencode.v30lift37.splicesites.txt -R -k../../../ genome_hg19/nochr -b
badreads.txt --rmIndels -o second
```

26 Collect filtered ALU, REP NON ALU and NON REP sites:

```
$ python../../../REDItools/NPscripts/collect_editing_candidates.py
$ sort -k1,1 -k2,2n editing.txt > editing_sorted.txt
```

The editing_sorted.txt file contains the list of RNA editing candidates for chromosome 4 of the NA12878 cell line. The full content is available online at https://github.com/BioinfoUNIBA/REDItools/tree/master/NPfiles.

27 Inspect the distribution of editing candidates to look at A-to-I enrichment:

```
$ python../../../REDItools/NPscripts/ get_Statistics.py
```

This script will create the editingStats.txt file (available at https://github.com/BioinfoUNIBA/REDItools/tree/master/NPfiles) including the fraction of detected substitutions.

Procedure 2: differential RNA editing in HD samples from BioProject PRJNA316625

Alignment of RNAseq reads from BioProject PRJNA316625 ● **Timing ~11 h**

1 Enter the PRJNA_316625 folder:

```
$ cd PRJNA_316625
```

2 Map each RNAseq sample onto the reference human genome using STAR:

```
$ cd SRR3306823
$ STAR --runThreadN 12 --genomeDir../../STAR/STAR_genome_index_ucsc/
--genomeLoad NoSharedMemory --readFilesIn SRR3306823_1.fastq.gz
SRR3306823_2.fastq.gz --readFilesCommand zcat --limitBAMsortRAM 0
--outFileNamePrefix SRR3306823_ --outReadsUnmapped Fastx --outSAMtype
BAM SortedByCoordinate --outSAMstrandField intronMotif --outSAMattri-
butes All --outFilterType BySJout --outFilterMultimapNmax 1 --outFilter
MismatchNmax 999 --outFilterMismatchNoverLmax 0.04 --alignIntronMin 20
--alignIntronMax 1000000 --alignMatesGapMax 1000000 --alignSJoverhang-
Min 8 --alignSJDBoverhangMin 1 --twopassMode Basic
$ mv SRR3306823_Aligned.sortedByCoord.out.bam SRR3306823.bam
$ samtools index SRR3306823.bam
$ cd..
$ cd SRR3306824
$ STAR --runThreadN 12 --genomeDir../../STAR/STAR_genome_index_ucsc/
--genomeLoad NoSharedMemory --readFilesIn SRR3306824_1.fastq.gz
SRR3306824_2.fastq.gz --readFilesCommand zcat --limitBAMsortRAM 0
--outFileNamePrefix SRR3306824_ --outReadsUnmapped Fastx --outSAMtype
BAM SortedByCoordinate --outSAMstrandField intronMotif --outSAMattri-
butes All --outFilterType BySJout --outFilterMultimapNmax 1 --outFilter
MismatchNmax 999 --outFilterMismatchNoverLmax 0.04 --alignIntronMin 20
--alignIntronMax 1000000 --alignMatesGapMax 1000000 --alignSJoverhang-
Min 8 --alignSJDBoverhangMin 1 --twopassMode Basic
$ mv SRR3306824_Aligned.sortedByCoord.out.bam SRR3306824.bam
$ samtools index SRR3306824.bam
$ cd..
$ cd SRR3306825
$ STAR --runThreadN 12 --genomeDir../../STAR/STAR_genome_index_ucsc/
--genomeLoad NoSharedMemory --readFilesIn SRR3306825_1.fastq.gz
SRR3306825_2.fastq.gz --readFilesCommand zcat --limitBAMsortRAM 0
--outFileNamePrefix SRR3306825_ --outReadsUnmapped Fastx --outSAMtype
BAM SortedByCoordinate --outSAMstrandField intronMotif --outSAMattri-
butes All --outFilterType BySJout --outFilterMultimapNmax 1 --outFilter
MismatchNmax 999 --outFilterMismatchNoverLmax 0.04 --alignIntronMin 20
```

```
--alignIntronMax 1000000 --alignMatesGapMax 1000000 --alignSJoverhang-
Min 8 --alignSJDBoverhangMin 1 --twopassMode Basic
$ mv SRR3306825_Aligned.sortedByCoord.out.bam SRR3306825.bam
$ samtools index SRR3306825.bam
$ cd..
$ cd SRR3306826
$ STAR --runThreadN 12 --genomeDir../../STAR/STAR_genome_index_ucsc/
--genomeLoad NoSharedMemory --readFilesIn SRR3306826_1.fastq.gz
SRR3306826_2.fastq.gz --readFilesCommand zcat --limitBAMsortRAM 0
--outFileNamePrefix SRR3306826_ --outReadsUnmapped Fastx --outSAMtype
BAM SortedByCoordinate --outSAMstrandField intronMotif --outSAMattri-
butes All --outFilterType BySJout --outFilterMultimapNmax 1 --outFilter
MismatchNmax 999 --outFilterMismatchNoverLmax 0.04 --alignIntronMin 20
--alignIntronMax 1000000 --alignMatesGapMax 1000000 --alignSJoverhang-
Min 8 --alignSJDBoverhangMin 1 --twopassMode Basic
$ mv SRR3306826_Aligned.sortedByCoord.out.bam SRR3306826.bam
$ samtools index SRR3306826.bam
$ cd..
$ cd SRR3306827
$ STAR --runThreadN 12 --genomeDir../../STAR/STAR_genome_index_ucsc/
--genomeLoad NoSharedMemory --readFilesIn SRR3306827_1.fastq.gz
SRR3306827_2.fastq.gz --readFilesCommand zcat --limitBAMsortRAM 0
--outFileNamePrefix SRR3306827_ --outReadsUnmapped Fastx --outSAMtype
BAM SortedByCoordinate --outSAMstrandField intronMotif --outSAMattri-
butes All --outFilterType BySJout --outFilterMultimapNmax 1 --outFilter
MismatchNmax 999 --outFilterMismatchNoverLmax 0.04 --alignIntronMin 20
--alignIntronMax 1000000 --alignMatesGapMax 1000000 --alignSJoverhang-
Min 8 --alignSJDBoverhangMin 1 --twopassMode Basic
$ mv SRR3306827_Aligned.sortedByCoord.out.bam SRR3306827.bam
$ samtools index SRR3306827.bam
$ cd..
$ cd SRR3306828
$ STAR --runThreadN 12 --genomeDir../../STAR/STAR_genome_index_ucsc/
--genomeLoad NoSharedMemory --readFilesIn SRR3306828_1.fastq.gz
SRR3306828_2.fastq.gz --readFilesCommand zcat --limitBAMsortRAM 0
--outFileNamePrefix SRR3306828_ --outReadsUnmapped Fastx --outSAMtype
BAM SortedByCoordinate --outSAMstrandField intronMotif --outSAMattri-
butes All --outFilterType BySJout --outFilterMultimapNmax 1 --outFilter
MismatchNmax 999 --outFilterMismatchNoverLmax 0.04 --alignIntronMin 20
--alignIntronMax 1000000 --alignMatesGapMax 1000000 --alignSJoverhang-
Min 8 --alignSJDBoverhangMin 1 --twopassMode Basic
$ mv SRR3306828_Aligned.sortedByCoord.out.bam SRR3306828.bam
$ samtools index SRR3306828.bam
$ cd..
$ cd SRR3306829
$ STAR --runThreadN 12 --genomeDir../../STAR/STAR_genome_index_ucsc/
--genomeLoad NoSharedMemory --readFilesIn SRR3306829_1.fastq.gz
SRR3306829_2.fastq.gz --readFilesCommand zcat --limitBAMsortRAM 0
--outFileNamePrefix SRR3306829_ --outReadsUnmapped Fastx --outSAMtype
BAM SortedByCoordinate --outSAMstrandField intronMotif --outSAMattri-
butes All --outFilterType BySJout --outFilterMultimapNmax 1 --outFilter
MismatchNmax 999 --outFilterMismatchNoverLmax 0.04 --alignIntronMin 20
--alignIntronMax 1000000 --alignMatesGapMax 1000000 --alignSJoverhang-
Min 8 --alignSJDBoverhangMin 1 --twopassMode Basic
$ mv SRR3306829_Aligned.sortedByCoord.out.bam SRR3306829.bam
$ samtools index SRR3306829.bam
$ cd..
$ cd SRR3306830
```

```
$ STAR --runThreadN 12 --genomeDir../../STAR/STAR_genome_index_ucsc/
--genomeLoad NoSharedMemory --readFilesIn SRR3306830_1.fastq.gz
SRR3306830_2.fastq.gz --readFilesCommand zcat --limitBAMsortRAM 0
--outFileNamePrefix SRR3306830_ --outReadsUnmapped Fastx --outSAMtype
BAM SortedByCoordinate --outSAMstrandField intronMotif --outSAMattri-
butes All --outFilterType BySJout --outFilterMultimapNmax 1 --outFilter
MismatchNmax 999 --outFilterMismatchNoverLmax 0.04 --alignIntronMin 20
--alignIntronMax 1000000 --alignMatesGapMax 1000000 --alignSJoverhang-
Min 8 --alignSJDBoverhangMin 1 --twopassMode Basic
$ mv SRR3306830_Aligned.sortedByCoord.out.bam SRR3306830.bam
$ samtools index SRR3306830.bam
$ cd..
$ cd SRR3306831
$ STAR --runThreadN 12 --genomeDir../../STAR/STAR_genome_index_ucsc/
--genomeLoad NoSharedMemory --readFilesIn SRR3306831_1.fastq.gz
SRR3306831_2.fastq.gz --readFilesCommand zcat --limitBAMsortRAM 0
--outFileNamePrefix SRR3306831_ --outReadsUnmapped Fastx --outSAMtype
BAM SortedByCoordinate --outSAMstrandField intronMotif --outSAMattri-
butes All --outFilterType BySJout --outFilterMultimapNmax 1 --outFilter
MismatchNmax 999 --outFilterMismatchNoverLmax 0.04 --alignIntronMin 20
--alignIntronMax 1000000 --alignMatesGapMax 1000000 --alignSJoverhang-
Min 8 --alignSJDBoverhangMin 1 --twopassMode Basic
$ mv SRR3306831_Aligned.sortedByCoord.out.bam SRR3306831.bam
$ samtools index SRR3306831.bam
$ cd..
$ cd SRR3306832
$ STAR --runThreadN 12 --genomeDir../../STAR/STAR_genome_index_ucsc/
--genomeLoad NoSharedMemory --readFilesIn SRR3306832_1.fastq.gz
SRR3306832_2.fastq.gz --readFilesCommand zcat --limitBAMsortRAM 0
--outFileNamePrefix SRR3306832_ --outReadsUnmapped Fastx --outSAMtype
BAM SortedByCoordinate --outSAMstrandField intronMotif --outSAMattri-
butes All --outFilterType BySJout --outFilterMultimapNmax 1 --outFilter
MismatchNmax 999 --outFilterMismatchNoverLmax 0.04 --alignIntronMin 20
--alignIntronMax 1000000 --alignMatesGapMax 1000000 --alignSJoverhang-
Min 8 --alignSJDBoverhangMin 1 --twopassMode Basic
$ mv SRR3306832_Aligned.sortedByCoord.out.bam SRR3306832.bam
$ samtools index SRR3306832.bam
$ cd..
$ cd SRR3306833
$ STAR --runThreadN 12 --genomeDir../../STAR/STAR_genome_index_ucsc/
--genomeLoad NoSharedMemory --readFilesIn SRR3306833_1.fastq.gz
SRR3306833_2.fastq.gz --readFilesCommand zcat --limitBAMsortRAM 0
--outFileNamePrefix SRR3306833_ --outReadsUnmapped Fastx --outSAMtype
BAM SortedByCoordinate --outSAMstrandField intronMotif --outSAMattri-
butes All --outFilterType BySJout --outFilterMultimapNmax 1 --outFilter
MismatchNmax 999 --outFilterMismatchNoverLmax 0.04 --alignIntronMin 20
--alignIntronMax 1000000 --alignMatesGapMax 1000000 --alignSJoverhang-
Min 8 --alignSJDBoverhangMin 1 --twopassMode Basic
$ mv SRR3306833_Aligned.sortedByCoord.out.bam SRR3306833.bam
$ samtools index SRR3306833.bam
$ cd..
$ cd SRR3306834
$ STAR --runThreadN 12 --genomeDir../../STAR/STAR_genome_index_ucsc/
--genomeLoad NoSharedMemory --readFilesIn SRR3306834_1.fastq.gz
SRR3306834_2.fastq.gz --readFilesCommand zcat --limitBAMsortRAM 0
--outFileNamePrefix SRR3306834_ --outReadsUnmapped Fastx --outSAMtype
BAM SortedByCoordinate --outSAMstrandField intronMotif --outSAMattri-
butes All --outFilterType BySJout --outFilterMultimapNmax 1
```

```
--outFilterMismatchNmax 999 --outFilterMismatchNoverLmax 0.04
--alignIntronMin 20 --alignIntronMax 1000000 --alignMatesGapMax
1000000 --alignSJoverhangMin 8 --alignSJDBoverhangMin 1 --twopassMode
Basic
$ mv SRR3306834_Aligned.sortedByCoord.out.bam SRR3306834.bam
$ samtools index SRR3306834.bam
$ cd..
$ cd SRR3306835
$ STAR --runThreadN 12 --genomeDir../../STAR/STAR_genome_index_ucsc/
--genomeLoad NoSharedMemory --readFilesIn SRR3306835_1.fastq.gz
SRR3306835_2.fastq.gz --readFilesCommand zcat --limitBAMsortRAM 0
--outFileNamePrefix SRR3306835_ --outReadsUnmapped Fastx --outSAMtype
BAM SortedByCoordinate --outSAMstrandField intronMotif --outSAMattri-
butes All --outFilterType BySJout --outFilterMultimapNmax 1 --outFilter
MismatchNmax 999 --outFilterMismatchNoverLmax 0.04 --alignIntronMin 20
--alignIntronMax 1000000 --alignMatesGapMax 1000000 --alignSJoverhang-
Min 8 --alignSJDBoverhangMin 1 --twopassMode Basic
$ mv SRR3306835_Aligned.sortedByCoord.out.bam SRR3306835.bam
$ samtools index SRR3306835.bam
$ cd..
$ cd SRR3306836
$ STAR --runThreadN 12 --genomeDir../../STAR/STAR_genome_index_ucsc/
--genomeLoad NoSharedMemory --readFilesIn SRR3306836_1.fastq.gz
SRR3306836_2.fastq.gz --readFilesCommand zcat --limitBAMsortRAM 0
--outFileNamePrefix SRR3306836_ --outReadsUnmapped Fastx --outSAMtype
BAM SortedByCoordinate --outSAMstrandField intronMotif --outSAMattri-
butes All --outFilterType BySJout --outFilterMultimapNmax 1 --outFilter
MismatchNmax 999 --outFilterMismatchNoverLmax 0.04 --alignIntronMin 20
--alignIntronMax 1000000 --alignMatesGapMax 1000000 --alignSJoverhang-
Min 8 --alignSJDBoverhangMin 1 --twopassMode Basic
$ mv SRR3306836_Aligned.sortedByCoord.out.bam SRR3306836.bam
$ samtools index SRR3306836.bam
$ cd..
```

3 Check the current working directory using `pwd`.
▲ **CRITICAL STEP** It must be PRJNA_316625.

Recovery of recoding RNA editing levels ● Timing ~50 min

4 Launch REDItoolDnaRna.py on each BAM file by limiting the analysis to recoding events only:

```
$ python../REDItools/main/REDItoolDnaRna.py -i SRR3306823/SRR3306823.
bam -o SRR3306823/editing/ -f../genome_hg19/GRCh37.primary_assembly.
genome.fa -c1,1 -m255,255 -v1 -q30,30 -e -n0.0 -N0.0 -u -l -p -s2 -g2 -S
-T../rediportal/srtd_atlas_recoding.gff
$ python../REDItools/main/REDItoolDnaRna.py -i SRR3306824/SRR3306824.
bam -o SRR3306824/editing/ -f../genome_hg19/GRCh37.primary_assembly.
genome.fa -c1,1 -m255,255 -v1 -q30,30 -e -n0.0 -N0.0 -u -l -p -s2 -g2 -S
-T../rediportal/srtd_atlas_recoding.gff
$ python../REDItools/main/REDItoolDnaRna.py -i SRR3306825/SRR3306825.
bam -o SRR3306825/editing/ -f../genome_hg19/GRCh37.primary_assembly.
genome.fa -c1,1 -m255,255 -v1 -q30,30 -e -n0.0 -N0.0 -u -l -p -s2 -g2 -S
-T../rediportal/srtd_atlas_recoding.gff
$ python../REDItools/main/REDItoolDnaRna.py -i SRR3306826/SRR3306826.
bam -o SRR3306826/editing/ -f../genome_hg19/GRCh37.primary_assembly.
genome.fa -c1,1 -m255,255 -v1 -q30,30 -e -n0.0 -N0.0 -u -l -p -s2 -g2 -S
-T../rediportal/srtd_atlas_recoding.gff
$ python../REDItools/main/REDItoolDnaRna.py -i SRR3306827/SRR3306827.
bam -o SRR3306827/editing/ -f../genome_hg19/GRCh37.primary_assembly.
```

```
genome.fa -c1,1 -m255,255 -v1 -q30,30 -e -n0.0 -N0.0 -u -l -p -s2 -g2 -S
-T../rediportal/srtd_atlas_recoding.gff
$ python../REDItools/main/REDItoolDnaRna.py -i SRR3306828/SRR3306828.
bam -o SRR3306828/editing/ -f../genome_hg19/GRCh37.primary_assembly.
genome.fa -c1,1 -m255,255 -v1 -q30,30 -e -n0.0 -N0.0 -u -l -p -s2 -g2 -S
-T../rediportal/srtd_atlas_recoding.gff
$ python../REDItools/main/REDItoolDnaRna.py -i SRR3306829/SRR3306829.
bam -o SRR3306829/editing/ -f../genome_hg19/GRCh37.primary_assembly.
genome.fa -c1,1 -m255,255 -v1 -q30,30 -e -n0.0 -N0.0 -u -l -p -s2 -g2 -S
-T../rediportal/srtd_atlas_recoding.gff
$ python../REDItools/main/REDItoolDnaRna.py -i SRR3306830/SRR3306830.
bam -o SRR3306830/editing/ -f../genome_hg19/GRCh37.primary_assembly.
genome.fa -c1,1 -m255,255 -v1 -q30,30 -e -n0.0 -N0.0 -u -l -p -s2 -g2 -S
-T../rediportal/srtd_atlas_recoding.gff
$ python../REDItools/main/REDItoolDnaRna.py -i SRR3306831/SRR3306831.
bam -o SRR3306831/editing/ -f../genome_hg19/GRCh37.primary_assembly.
genome.fa -c1,1 -m255,255 -v1 -q30,30 -e -n0.0 -N0.0 -u -l -p -s2 -g2 -S
-T../rediportal/srtd_atlas_recoding.gff
$ python../REDItools/main/REDItoolDnaRna.py -i SRR3306832/SRR3306832.
bam -o SRR3306832/editing/ -f../genome_hg19/GRCh37.primary_assembly.
genome.fa -c1,1 -m255,255 -v1 -q30,30 -e -n0.0 -N0.0 -u -l -p -s2 -g2 -S
-T../rediportal/srtd_atlas_recoding.gff
$ python../REDItools/main/REDItoolDnaRna.py -i SRR3306833/SRR3306833.
bam -o SRR3306833/editing/ -f../genome_hg19/GRCh37.primary_assembly.
genome.fa -c1,1 -m255,255 -v1 -q30,30 -e -n0.0 -N0.0 -u -l -p -s2 -g2 -S
-T../rediportal/srtd_atlas_recoding.gff
$ python../REDItools/main/REDItoolDnaRna.py -i SRR3306834/SRR3306834.
bam -o SRR3306834/editing/ -f../genome_hg19/GRCh37.primary_assembly.
genome.fa -c1,1 -m255,255 -v1 -q30,30 -e -n0.0 -N0.0 -u -l -p -s2 -g2 -S
-T../rediportal/srtd_atlas_recoding.gff
$ python../REDItools/main/REDItoolDnaRna.py -i SRR3306835/SRR3306835.
bam -o SRR3306835/editing/ -f../genome_hg19/GRCh37.primary_assembly.
genome.fa -c1,1 -m255,255 -v1 -q30,30 -e -n0.0 -N0.0 -u -l -p -s2 -g2 -S
-T../rediportal/srtd_atlas_recoding.gff
$ python../REDItools/main/REDItoolDnaRna.py -i SRR3306836/SRR3306836.
bam -o SRR3306836/editing/ -f../genome_hg19/GRCh37.primary_assembly.
genome.fa -c1,1 -m255,255 -v1 -q30,30 -e -n0.0 -N0.0 -u -l -p -s2 -g2 -S
-T../rediportal/srtd_atlas_recoding.gff
```

Detection of differential RNA editing ● Timing ~5 min

5 Create a comma-separated information file including sample names and the diseased status (DIS: diseased; CTRL: control), according to BioProject PRJNA316625. Any text editor can be used. The file content has the following structure:

```
Sample,Status
SRR3306823,DIS
SRR3306824,DIS
SRR3306825,DIS
SRR3306826,DIS
SRR3306827,DIS
SRR3306828,DIS
SRR3306829,DIS
SRR3306830,CTRL
SRR3306831,CTRL
SRR3306832,CTRL
SRR3306833,CTRL
SRR3306834,CTRL
```

SRR3306835,CTRL
SRR3306836,CTRL

A copy of the sample information file is available online at https://github.com/BioinfoUNIBA/REDItools/tree/master/NPfiles.

6 Detect differential RNA editing:

```
$ python../REDItools/accessory/get_DE_events.py
-cpval 2 -input_file sample_information.csv -sig yes
```

A copy of the output table is available online at https://github.com/BioinfoUNIBA/REDItools/tree/master/NPfiles.

For control case studies by launching the `get_DE_events.py` script, the user can filter `REDItoolDnaRna.py` outputs according to the following criteria:

- RNAseq coverage per position (default 10 reads)
- Minimum editing frequency per position (default 10%)

For each editing candidate, the script applies the Mann–Whitney test to check the significance between the two conditions, control and HD. By default, the test is carried out only if the number of editing events per position is equal to 50% of the samples per group.

Optionally, *P* values can be corrected using Benjamini–Hochberg or Bonferroni tests.

Users can also detect differential RNA editing using the linear model implemented in Tran et al.[74] by using the option `-linear`.

Troubleshooting

Troubleshooting advice can be found in Table 2.

Table 2 | Troubleshooting table

Step	Problem	Possible reason	Solution
8 (Procedure 1)	REDItools reports that it cannot open input files.	WGS or RNAseq BAMs are not correctly indexed, or paths to input files are not correct.	Check and fix the indexing of BAM files and input paths.
11 (Procedure 1)	AnnotateTable.py reports an error message that input files cannot be opened.	Annotation files are not correctly indexed, or their paths are not correct.	Check the indexing of annotation files and input paths.
23 (Procedure 1)	pblat returns an execution error.	Input file with reads is empty.	Skip this step and create an empty badreads.txt file.
24 (Procedure 1)	SAMtools return an execution error.	Input bed file is empty.	Skip this step because no reads need to be cleaned.

Timing

Both procedures have been run on a machine with 40 cores, 256 GB of RAM and several TB of disk space. The time spent depends on how many resources can be allocated. On average, both procedures should take ~76 h using 4–24 cores. The bottleneck is the alignment of WGS reads.

Procedure 1:

Steps 1–4, preprocessing and alignment of RNAseq reads onto the genome (4 cores): ~1 h
Step 5, detection of the strand orientation of RNAseq reads: ~1 min
Steps 6 and 7, alignment of WGS reads onto the genome (24 cores): ~30 h
Step 8, identification of all DNA–RNA variants (first REDItools round): ~3 h
Steps 9–27, filtering of DNA–RNA variants: ~30 h

Procedure 2:

Steps 1–3, alignment of RNAseq reads from BioProject PRJNA316625: ~11 h
Step 4, recovering of recoding RNA editing levels: ~50 min
Steps 5 and 6, detection of differential RNA editing: ~5 min

Table 3 | Distribution of observed substitutions calculated globally (ALL) and per each category of sites: ALU, REP NON ALU, NON REP, coding and noncoding

SubType	ALU	REP NON ALU	NON REP	Coding	Noncoding	ALL
AC	0.24	0.0	0.0	0.0	0.24	0.24
GT	0.57	0.0	0.0	0.0	0.56	0.56
AG	92.49	0.0	100.0	100.0	92.71	92.64
CA	0.33	0.0	0.0	0.0	0.32	0.32
CG	0.24	0.0	0.0	0.0	0.24	0.24
GC	0.49	0.0	0.0	0.0	0.48	0.48
AT	0.57	0.0	0.0	0.0	0.56	0.56
GA	1.22	0.0	0.0	0.0	1.20	1.20
TG	0.24	0.0	0.0	0.0	0.24	0.24
CT	1.55	0.0	0.0	0.0	1.52	1.52
TC	1.88	0.0	0.0	0.0	1.84	1.84
TA	0.16	0.0	0.0	0.0	0.16	0.16

Anticipated results

Procedure 1: RNA editing detection in chromosome 4 of the NA12878 cell line

Following the above procedure, the first round of REDItoolDnaRna.py yields 5,765,292 RNA-DNA variants covered by at least one RNAseq read. This number decreases to only 15,041 sites when invariant RNA positions and changes not supported by ≥10 WGS reads are removed. After the annotation steps and further filters, users should obtain 679 editing candidates. Of these, 421 changes are already known to be edited according to our REDIportal database. The remaining sites are subjected to stringent filters to eliminate further errors, and then all cleaned positions are collected and returned as RNA editing candidates. A simple control to verify the reliability of predictions consists of calculating the distribution of all observed substitutions to look at an A-to-G enrichment corresponding to potential A-to-I editing events. As shown in Table 3, ~93% of all detected changes are A to G, confirming the accuracy of predictions and the validity of the bioinformatics pipeline. This conclusion is further supported by the fact that >80% of detected candidates are already known as A-to-I editing events. Results have also been refined by annotating A-to-G candidates using ANNOVAR[75,76] and Gencode genes (Table 3). The majority of events reside in non-coding RNAs (33.6%) or introns (20.8%) or 3′ UTRs (36.4%), while only a very tiny fraction is located in the protein-coding region (0.23%). Although the protocol as shown here is limited to the discovery of RNA editing events in chromosome 4, results are perfectly in line with previous studies in humans, in which edited sites mainly reside in intronic sequences rich in Alu repetitive elements and a only a small fraction occurs in coding sequences[10,77].

Procedure 2: differential RNA editing in HD

According to our pipeline, PRJNA316625 raw reads are cleaned and aligned onto the human reference genome. Next, REDItoolDnaRna.py is launched on a collection of known recoding editing events (including 1,902 sites) from our REDIportal database. Dysregulated positions are detected by calculating per each site the Mann–Whitney *P* value if covered in ≥50% of samples per group. At the end, users should recover 39 positions with a valid Mann–Whitney *P* value. Of these, five sites are potentially dysregulated at 0.05 significance level (Table 4) and are located in *GRIK2*, *GRIA2*, *NOVA1* and *CABP1* protein-coding genes. While *GRIK2* and *GRIA2* belong to glutamate ionotropic receptors and are activated in various normal neurophysiologic processes, NOVA1 is an alternative splicing regulator that may modulate RNA splicing at different levels. CABP1 is a calcium-binding protein that may be relevant in calcium-mediated cellular signal transduction. It is interesting to note that at selected positions, RNA editing tends to decrease in HD samples. Reduced editing levels at recoding positions have also been observed in other neurodegenerative disorders[37].

Users requiring more stringent filters and working with thousand of positions could apply a multiple testing correction like Benjamini–Hochberg or Bonferroni. In addition, interested users could calculate differential RNA editing using the methodology by Tran et al.[74] implemented in our script `get_DE_events.py` at https://github.com/BioinfoUNIBA/REDItools/tree/master/NPfiles.

Table 4 | Potential dysregulated RNA editing events in HD samples from BioProject PRJNA316625

Position (hg19)	Gene name	AA change	Controls	HDs	*P* value
chr6:102372589	*GRIK2*	Q621R	0.904	0.844	0.017
chr6:102337702	*GRIK2*	Y571C	0.846	0.777	0.020
chr4:158281294	*GRIA2*	R717G	0.591	0.497	0.029
chr14:26917530	*NOVA1*	S265G	0.403	0.307	0.025
chr12:121078907	*CABP1*	Q140R	0.307	0.233	0.008

We report the genomic position (human genome assembly hg19), the gene name, the amino acid (AA) change induced by RNA editing, the mean editing level in controls, the mean editing level in HD samples and the Mann–Whitney *P* value.

Reporting Summary

Further information on research design is available in the Nature Research Reporting Summary linked to this article.

Data availability

Example datasets that include NA12878 RNAseq/WGS reads and RNAseq reads from BioProject PRJNA316625 are freely available at the ENA database (https://www.ebi.ac.uk/ena) or the SRA archive (https://www.ncbi.nlm.nih.gov/sra).

Code availability

REDItools source code is available at the GitHub website (https://github.com/BioinfoUNIBA/REDItools) under the MIT License. The code in this protocol has been peer reviewed.

References

1. Saletore, Y., Meyer, K., Korlach, J., Vilfan, I. D., Jaffrey, S. & Mason, C. E. The birth of the Epitranscriptome: deciphering the function of RNA modifications. *Genome Biol.* **13**, 175 (2012).
2. Boccaletto, P. et al. MODOMICS: a database of RNA modification pathways. 2017 update. *Nucleic Acids Res.* **46**, D303–D307 (2018).
3. Jantsch, M. F. & Schaefer, M. R. Mining the epitranscriptome: detection of RNA editing and RNA modifications. *Methods* **156**, 1–4 (2019).
4. Benne, R. et al. Major transcript of the frameshifted *coxII* gene from trypanosome mitochondria contains four nucleotides that are not encoded in the DNA. *Cell* **46**, 819–826 (1986).
5. Gott, J. M. & Emeson, R. B. Functions and mechanisms of RNA editing. *Annu. Rev. Genet.* **34**, 499–531 (2000).
6. Eisenberg, E. & Levanon, E. Y. A-to-I RNA editing—immune protector and transcriptome diversifier. *Nat. Rev. Genet.* **19**, 473–490 (2018).
7. Rosenberg, B. R., Hamilton, C. E., Mwangi, M. M., Dewell, S. & Papavasiliou, F. N. Transcriptome-wide sequencing reveals numerous APOBEC1 mRNA-editing targets in transcript 3' UTRs. *Nat. Struct. Mol. Biol.* **18**, 230–236 (2011).
8. Nishikura, K. A-to-I editing of coding and non-coding RNAs by ADARs. *Nat. Rev. Mol. Cell Biol.* **17**, 83–96 (2016).
9. Bazak, L. et al. A-to-I RNA editing occurs at over a hundred million genomic sites, located in a majority of human genes. *Genome Res.* **24**, 365–376 (2014).
10. Picardi, E. et al. Profiling RNA editing in human tissues: towards the inosinome Atlas. *Sci. Rep.* **5**, 14941 (2015).
11. Mallela, A. & Nishikura, K. A-to-I editing of protein coding and noncoding RNAs. *Crit. Rev. Biochem. Mol. Biol.* **47**, 493–501 (2012).
12. Mannion, N. M. et al. The RNA-editing enzyme ADAR1 controls innate immune responses to RNA. *Cell Rep.* **9**, 1482–1494 (2014).
13. Gallo, A., Vukic, D., Michalik, D., O'Connell, M. A. & Keegan, L. P. ADAR RNA editing in human disease; more to it than meets the I. *Hum. Genet.* **136**, 1265–1278 (2017).
14. Silvestris, D. A. et al. Dynamic inosinome profiles reveal novel patient stratification and gender-specific differences in glioblastoma. *Genome Biol.* **20**, 33 (2019).
15. Ramaswami, G. et al. Accurate identification of human Alu and non-Alu RNA editing sites. *Nat. Methods* **9**, 579–581 (2012).
16. Eisenberg, E. Bioinformatic approaches for identification of A-to-I editing sites. *Curr. Top. Microbiol. Immunol.* **353**, 145–162 (2012).

17. Diroma, M. A., Ciaccia, L., Pesole, G. & Picardi, E. Elucidating the editome: bioinformatics approaches for RNA editing detection. *Brief. Bioinform.* **20**, 436–447 (2019).
18. Nigita, G., Alaimo, S., Ferro, A., Giugno, R. & Pulvirenti, A. Knowledge in the investigation of A-to-I RNA editing signals. *Front. Bioeng. Biotechnol.* **3**, 18 (2015).
19. Nigita, G. et al. ncRNA editing: functional characterization and computational resources. in *Computational Biology of Non-Coding RNA* (eds Lai, X., Gupta, S. K. & Vera, J.) 133–174 (Humana Press, 2019).
20. Picardi, E. & Pesole, G. REDItools: high-throughput RNA editing detection made easy. *Bioinformatics* **29**, 1813–1814 (2013).
21. Picardi, E., D'Erchia, A. M., Lo Giudice, C. & Pesole, G. REDIportal: a comprehensive database of A-to-I RNA editing events in humans. *Nucleic Acids Res.* **45**, D750–D757 (2017).
22. 1000 Genomes Project Consortium. A global reference for human genetic variation. *Nature* **526**, 68 (2015).
23. Chen, S., Zhou, Y., Chen, Y. & Gu, J. fastp: an ultra-fast all-in-one FASTQ preprocessor. *Bioinformatics* **34**, 6 (2018).
24. Dobin, A. et al. STAR: ultrafast universal RNA-seq aligner. *Bioinformatics* **29**, 15–21 (2013).
25. Li, H. & Durbin, R. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics* **25**, 1754–1760 (2009).
26. Li, H. et al. The Sequence Alignment/Map format and SAMtools. *Bioinformatics* **25**, 2078–2079 (2009).
27. Wang, M. & Kong, L. pblat: a multithread blat algorithm speeding up aligning sequences to genomes. *BMC Bioinformatics* **20**, 28 (2019).
28. Kent, W. J. BLAT-the BLAST-like alignment tool. *Genome Res.* **12**, 656–664 (2002).
29. Ross, C. A. & Tabrizi, S. J. Huntington's disease: from molecular pathogenesis to clinical treatment. *Lancet Neurol.* **10**, 15 (2011).
30. Hodges, A. et al. Regional and cellular gene expression changes in human Huntington's disease brain. *Hum. Mol. Genet.* **15**, 965–977 (2006).
31. Valor, L. M. Transcription, epigenetics and ameliorative strategies in Huntington's Disease: a genome-wide perspective. *Mol. Neurobiol.* **51**, 406–423 (2015).
32. Marti, E. et al. A myriad of miRNA variants in control and Huntington's disease brain regions detected by massively parallel sequencing. *Nucleic Acids Res.* **38**, 7219–7235 (2010).
33. Luthi-Carter, R. et al. Decreased expression of striatal signaling genes in a mouse model of Huntington's disease. *Hum. Mol. Genet.* **9**, 1259–1271 (2000).
34. Lin, L. et al. Transcriptome sequencing reveals aberrant alternative splicing in Huntington's disease. *Hum. Mol. Genet.* **25**, 3454–3466 (2016).
35. Annese, A. et al. Whole transcriptome profiling of late-onset Alzheimer's disease patients provides insights into the molecular changes involved in the disease. *Sci. Rep.* **8**, 4282 (2018).
36. D'Erchia, A. M. et al. Massive transcriptome sequencing of human spinal cord tissues provides new insights into motor neuron degeneration in ALS. *Sci. Rep.* **7**, 10046 (2017).
37. Khermesh, K. et al. Reduced levels of protein recoding by A-to-I RNA editing in Alzheimer's disease. *RNA* **22**, 290–302 (2016).
38. Srivastava, P. K. et al. Genome-wide analysis of differential RNA editing in epilepsy. *Genome Res.* **27**, 440–450 (2017).
39. Wang, L., Wang, S. & Li, W. RSeQC: quality control of RNA-seq experiments. *Bioinformatics* **28**, 2184–2185 (2012).
40. DeLuca, D. S. et al. RNA-SeQC: RNA-seq metrics for quality control and process optimization. *Bioinformatics* **28**, 1530–1532 (2012).
41. Bolger, A. M., Lohse, M. & Usadel, B. Trimmomatic: a flexible trimmer for Illumina sequence data. *Bioinformatics* **30**, 2114–2120 (2014).
42. Martin, M. Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet J.* **17**, 10–12 (2011).
43. Patel, R. K. & Jain, M. NGS QC Toolkit: a toolkit for quality control of next generation sequencing data. *PlOS One* **7**, e30619 (2012).
44. Wu, T. D. & Nacu, S. Fast and SNP-tolerant detection of complex variants and splicing in short reads. *Bioinformatics* **26**, 873–881 (2011).
45. Kim, D., Langmead, B. & Salzberg, S. L. HISAT: a fast spliced aligner with low memory requirements. *Nat. Methods* **12**, 357–360 (2015).
46. Langmead, B. & Salzberg, S. L. Fast gapped-read alignment with Bowtie 2. *Nat. Methods* **9**, 2 (2012).
47. Yu, C. et al. SOAP2: an improved ultrafast tool for short read alignment. *Bioinformatics* **25**, 2 (2009).
48. John, D., Weirick, T., Dimmeler, S. & Uchida, S. RNAEditor: easy detection of RNA editing events and the introduction of editing islands. *Brief. Bioinform.* **18**, 8 (2012).
49. Wang, Z. et al. RES-Scanner: a software package for genome-wide identification of RNA-editing sites. *Gigascience* **5**, 37 (2016).
50. Zhang, Q. & Xiao, X. Genome sequence-independent identification of RNA editing sites. *Nat. Methods* **12**, 347 (2015).
51. Piechotta, M., Wyler, E., Ohler, U., Landthaler, M. & Dieterich, C. JACUSA: site-specific identification of RNA editing events from replicate sequencing data. *BMC Bioinform.* **18**, 7 (2017).

52. Kim, M. S., Hur, B. & Kim, S. RDDpred: a condition-specific RNA-editing prediction model from RNA-seq data. BMC Genomics 17(Suppl 1)), 5 (2016).
53. Xiong, H. et al. RED-ML: a novel, effective RNA editing detection method based on machine learning. *Gigascience* https://doi.org/10.1093/gigascience/gix012 (2017).
54. Ouyang, Z. et al. Accurate identification of RNA editing sites from primitive sequence with deep neural networks. *Sci. Rep.* **8**, 6005 (2018).
55. Porath, H. T., Carmi, S. & Levanon, E. Y. A genome-wide map of hyper-edited RNA reveals numerous new sites. *Nat. Commun.* **5**, 4726 (2014).
56. Zhang, F., Lu, Y., Yan, S., Xing, Q. & Tian, W. SPRINT: an SNP-free toolkit for identifying RNA editing sites. *Bioinformatics* **33**, 3538–3548 (2017).
57. Kiran, A. M., O'Mahony, J. J., Sanjeev, K. & Baranov, P. V. Darned in 2013: inclusion of model organisms and linking with Wikipedia. *Nucleic Acids Res.* **41**, D258–261 (2013).
58. Ramaswami, G. & Li, J. B. RADAR: a rigorously annotated database of A-to-I RNA editing. *Nucleic Acids Res.* **42**, D109–113 (2014).
59. Picardi, E., Horner, D. S. & Pesole, G. Single cell transcriptomics reveals specific RNA editing signatures in the human brain. *RNA* **23**, 860–865 (2017).
60. Rossetti, C. et al. RNA editing signature during myeloid leukemia cell differentiation. *Leukemia* **31**, 2824–2832 (2017).
61. Pinto, Y., Buchumenski, I., Levanon, E. Y. & Eisenberg, E. Human cancer tissues exhibit reduced A-to-I editing of miRNAs coupled with elevated editing of their targets. *Nucleic Acids Res.* **46**, 71–82 (2018).
62. Lin, C.-H. & Chen, S. C.-C. The Cancer Editome Atlas: a resource for exploratory analysis of the adenosine-to-inosine RNA editome in cancer. *Cancer Res.* **79**, 3001–3006 (2019).
63. Porath, H. T. et al. RNA editing is abundant and correlates with task performance in a social bumblebee. *Nat. Commun.* **10**, 1605 (2019).
64. Liu, H. et al. A-to-I RNA editing is developmentally regulated and generally adaptive for sexual reproduction in *Neurospora crassa*. *Proc. Natl Acad. Sci. USA* **114**, E7756–E7765 (2017).
65. Liew, Y. J., Li, Y., Baumgarten, S., Voolstra, C. R. & Aranda, M. Condition-specific RNA editing in the coral symbiont *Symbiodinium microadriaticum*. *PLOS Genet.* **13**, e1006619 (2017).
66. Sapiro, A. L. et al. Illuminating spatial A-to-I RNA editing signatures within the *Drosophila* brain. *Proc. Natl Acad. Sci. USA* **116**, 2318–2327 (2019).
67. Picardi, E. et al. Large-scale detection and analysis of RNA editing in grape mtDNA by RNA deep-sequencing. *Nucleic Acids Res.* **38**, 4755–4767 (2010).
68. Wu, B. et al. Identification of symmetrical RNA editing events in the mitochondria of *Salvia miltiorrhiza* by strand-specific RNA sequencing. *Sci. Rep.* **7**, 42250 (2017).
69. Picardi, E., D'Erchia, A. M., Montalvo, A. & Pesole, G. Using REDItools to detect RNA editing events in NGS datasets. *Curr. Protoc. Bioinforma.* **49**, 12.12.1–12.12.15 (2015).
70. Picardi, E., D'Erchia, A. M., Gallo, A., Montalvo, A. & Pesole, G. Uncovering RNA editing sites in long non-coding RNAs. *Front. Bioeng. Biotechnol.* **2**, 64 (2014).
71. Pinto, Y., Cohen, H. Y. & Levanon, E. Y. Mammalian conserved ADAR targets comprise only a small fragment of the human editosome. *Genome Biol.* **15**, R5 (2014).
72. Deininger, P. Alu elements: know the SINEs. *Genome Biol.* **12**, 236 (2011).
73. Porath, H. T., Knisbacher, B. A., Eisenberg, E. & Levanon, E. Y. Massive A-to-I RNA editing is common across the Metazoa and correlates with dsRNA abundance. *Genome Biol.* **18**, 185 (2017).
74. Tran, S. S. et al. Widespread RNA editing dysregulation in brains from autistic individuals. *Nat. Neurosci.* **22**, 25 (2019).
75. Wang, K., Li, M. & Hakonarson, H. ANNOVAR: functional annotation of genetic variants from high-throughput sequencing data. *Nucleic Acids Res.* **38**, e164 (2010).
76. Yang, H. & Wang, K. Genomic variant annotation and prioritization with ANNOVAR and wANNOVAR. *Nat. Protoc.* **10**, 1556 (2015).
77. Tan, M. H. et al. Dynamic landscape and regulation of RNA editing in mammals. *Nature* **550**, 249 (2017).

Acknowledgements

We kindly thank the ReCaS computing center at University of Bari for computational and technical assistance. We also acknowledge the PRACE project 2016163924 for computing resources. This work was supported by Elixir IIB and PRACE projects 2016163924 and 2018194670.

Author contributions

G.P. and E.P. conceived the project. C.L.G. and E.P. designed and wrote the full analysis pipeline. C.L.G. performed extensive user testing of the software. M.A.T. created the docker image and implemented the statistical tests to calculate differential RNA editing. E.P. wrote the manuscript with input and editing from all authors.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information is available for this paper at https://doi.org/10.1038/s41596-019-0279-7.

Correspondence and requests for materials should be addressed to E.P.

Peer review information *Nature Protocols* thanks Trees-Juen Chuang, Erez Levanon and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.

Received: 18 June 2019; Accepted: 5 December 2019;
Published online: 29 January 2020

Related links

Key references using this protocol

D'Erchia, A. M. et al. *Sci. Rep.* **7**, 10046 (2017): https://doi.org/10.1038/s41598-017-10488-7
Picardi, E., Horner, D. S. and Pesole, G. *RNA* **23**, 860–865 (2017): https://doi.org/10.1261/rna.058271.116
Rossetti, C. et al. *Leukemia* **31**, 2824–2832 (2017): https://doi.org/10.1038/leu.2017.134

nature research

Corresponding author(s): Ernesto Picardi

Last updated by author(s): Sep 10, 2019

Reporting Summary

Nature Research wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Research policies, see Authors & Referees and the Editorial Policy Checklist.

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

n/a	Confirmed
☒	☐ The exact sample size (*n*) for each experimental group/condition, given as a discrete number and unit of measurement
☒	☐ A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
☐	☒ The statistical test(s) used AND whether they are one- or two-sided *Only common tests should be described solely by name; describe more complex techniques in the Methods section.*
☒	☐ A description of all covariates tested
☐	☒ A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
☐	☒ A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
☒	☐ For null hypothesis testing, the test statistic (e.g. *F*, *t*, *r*) with confidence intervals, effect sizes, degrees of freedom and *P* value noted *Give P values as exact values whenever suitable.*
☒	☐ For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
☒	☐ For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
☒	☐ Estimates of effect sizes (e.g. Cohen's *d*, Pearson's *r*), indicating how they were calculated

Our web collection on statistics for biologists contains articles on many of the points above.

Software and code

Policy information about availability of computer code

Data collection	All data used in this study are freely available at ENA (https://www.ebi.ac.uk/ena) or SRA databases (https://www.ncbi.nlm.nih.gov/sra). They include: 1) WGS data for cell line NA12878 (ERR262997); 2) RNAseq data for cell line NA12878 (SRR1258218); 3) BioProject PRJNA316625.
Data analysis	Conda environment manager (https://docs.conda.io/en/latest/miniconda.html) and the following packages: bcftools, bedtools, bwa, bx-python, fastp, fastqc, fisher, gmap, htslib, libdeflate, pysam, rseqc, samtools, star, bzip2, git, numpy, scipy, wget. REDItools 1.3 and accessory scripts: python scripts developed with the aim to study RNA editing at genomic scale by next generation sequencing data (https://github.com/BioinfoUNIBA/REDItools). Pblat: parallelized blat with multi-threads support (http://icebert.github.io/pblat/)

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors/reviewers. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Research guidelines for submitting code & software for further information.

Data

Policy information about availability of data

All manuscripts must include a data availability statement. This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A list of figures that have associated raw data
- A description of any restrictions on data availability

Example datasets that include NA12878 RNAseq/WGS reads and RNAseq reads from BioProject PRJNA316625 are freely available at ENA database (https://www.ebi.ac.uk/ena) or SRA archive (https://www.ncbi.nlm.nih.gov/sra).

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

☒ Life sciences ☐ Behavioural & social sciences ☐ Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/documents/nr-reporting-summary-flat.pdf

Life sciences study design

All studies must disclose on these points even when the disclosure is negative.

Sample size	BioProject PRJNA316625 including 14 samples, 7 controls and 7 affected by Huntington's disease.
Data exclusions	No data has been excluded.
Replication	Each group in BioProject PRJNA316625 includes biological replicates.
Randomization	Not relevant for this study.
Blinding	Not relevant for this study.

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems

n/a	Involved in the study
☒	☐ Antibodies
☒	☐ Eukaryotic cell lines
☒	☐ Palaeontology
☒	☐ Animals and other organisms
☒	☐ Human research participants
☒	☐ Clinical data

Methods

n/a	Involved in the study
☒	☐ ChIP-seq
☒	☐ Flow cytometry
☒	☐ MRI-based neuroimaging