Supplementary Methods

"Sex-Aware Genome-Wide Assessment of De Novo Variants in Autism Across Coding and Noncoding Regions"

Tychele N. Turner

SUPPLEMENTARY METHODS

HAT-FLEX Detailed Method

HAT-FLEX (https://github.com/TNTurnerLab/HAT-FLEX) is a Python-based post-calling pipeline designed to identify and curate high-confidence DNVs from parent-offspring trios. It operates on one or two per-trio genome-wide VCFs generated by different variant callers (e.g., GATK (1), DeepVariant (2, 3)) and applies sex-aware haploid logic on the sex chromosomes, utilizes several variant quality filters, can optionally mask user-provided genomic regions, annotates variant clustering, and writes a sorted, compressed VCF plus optional summary files for each child. While its initial design focused on human data, it can also run on data from other diploid species (e.g., mouse).

For each trio, HAT-FLEX takes as input at least one genome-wide trio VCF and optionally a second caller VCF. Each of these can be either a single trio-level VCF or optionally VCFs containing additional individuals can also be used by HAT-FLEX. If both callers are provided, HAT-FLEX runs in two-caller intersection mode. If only one VCF is provided, it instead runs in caller1-only mode for that child and records this behavior in the logs.

Trio relationships can be defined in two ways as input files. The first way is by generating a simple family file that consists of plain text lines of the form fatherID,motherID,childID,[childSex], where fields are separated by commas or whitespace. Providing the sex of the child is optional; however, HAT-FLEX will not consider the sex chromosomes if not given. The second way is to provide a standard pedigree file with columns FID,IID,PID,MID,SEX,PHENO; HAT-FLEX identifies complete trios by selecting individuals whose PID and MID are both non-zero and also defined as individuals in the file. In this format, SEX=1 indicates a male, SEX=2 indicates a female, and other or missing values are treated as sex-unknown. When both a family file and a pedigree file are provided, trios from both sources are merged per child ID, and the user specifies which source is preferred. The default is to use the pedigree. If there are conflicting parents or child sex between sources, the preferred source is kept and the alternative source is used only to fill in missing sex values when they are not contradictory. If sex is not specified in either file, the user may set the sex of the child as male or female with a flag. When neither explicit nor default sex is available, the child is treated as sex-unknown, only autosomes (chromosomes 1 to 22) are analyzed, and the X chromosome and Y chromosome sites are skipped. If no valid trios are identified from either source, the program terminates with an error.

To determine sex-aware ploidy on the X chromosome and the Y chromosome, HAT-FLEX uses pseudoautosomal region (PAR) coordinates. These can be supplied as a BED or BED.gz file; if a valid file is not provided, built-in hg38 defaults are used by HAT-FLEX. For the X chromosome, PAR1 is defined from 10,000 bp to 2,781,479 bp and PAR2 from 155,701,382 bp to 156,030,895 bp; for the Y chromosome, PAR1 is from 10,000 bp to 2,781,479 bp and PAR2 is from 56,887,902 bp to 57,217,415 bp. Internally, BED coordinates are treated as 0-based, half-open intervals and converted to 1-based inclusive intervals by adding 1 to the start coordinate and using the end coordinate as given. Chromosome names with or without the “chr” prefix are treated equivalently. Using these regions, HAT-FLEX determines haploid vs diploid context for each site: autosomes are always diploid; sex-unknown children have all X chromosome and Y chromosome loci skipped; male non-PAR X chromosome loci are treated as haploid X; male non-PAR Y chromosome loci are treated as haploid Y; and PAR loci are always treated as diploid regardless of sex. This ploidy state controls which parents are considered relevant at each locus, the applicable depth thresholds, allele balance criteria, and whether a site is flagged as “haploid unusual.” For each trio VCF, HAT-FLEX scans the file and identifies candidate *de novo* loci based on simple genotype patterns in the trio samples. The VCF header #CHROM line is used to map sample names to column indices, and if any of the father, mother, or child IDs are missing, the program quits for that trio. The FORMAT field is parsed, and the GT subfield is required; missing or partial genotypes such as ./., .|.,

or . are treated as missing. Genotype alleles are converted to integer indices, with 0 representing the reference allele and values greater than zero representing alternate alleles. At this stage, HAT-FLEX applies genotype-based criteria to flag *de novo* candidate loci at the site level. On autosomes, a locus is considered *de novo* if both parents are homozygous reference (0/0) and the child is non-reference (i.e., carries any alternate allele). On the male non-PAR X chromosome, the mother must be homozygous reference, and the child must be non-reference; on the male non-PAR Y chromosome, the father must be homozygous reference, and the child must be non-reference. Sites on the X chromosome and Y chromosome are skipped entirely if the child's sex is unknown. Sites that do not meet these patterns of genotypes are not considered *de novo* and are dropped from further analysis.

HAT-FLEX supports two modes of handling multi-allelic sites: allele-level mode (allele, the default) and locus-level mode (locus). In allele mode, multi-allelic sites are split to consider individual alternate alleles separately, and a candidate *de novo* is recorded only for specific ALT alleles that are present in the child's genotype. Symbolic alleles such as <NON_REF>, <*>, or breakend-style alleles are excluded from *de novo* consideration. Each candidate allele is indexed by a key of the form chrom:norm_pos:norm_ref:norm_alt, where normalized sequences are used. In locus mode, the site is treated as a single locus, regardless of multiple alternate alleles; an arbitrary first alternate allele is used to construct the key and alt_index is set to −1. Normalization of alleles, controlled by --normalize-alleles, trims shared prefix bases while incrementing the position accordingly, trims shared suffix bases, and ensures that both the reference allele and the alternate allele retain at least one nucleotide.

In dual-caller intersection mode, candidate *de novo* loci are identified independently from the caller1 and caller2 VCFs for each child, with each producing a mapping from the normalized key to the VCF row and allele index. Only keys present in both callers are retained as final candidates, so the set intersection of keys defines the *de novo* candidates that proceed to full filtering and annotation. In caller1-only mode, all keys from caller1 are used as the candidate set. Region masking can be applied to these candidates by providing either a directory containing .bed and/or .bed.gz files or a tar/tar.gz archive of BED files. Each BED entry is parsed as a 0-based, half-open interval (chrom start end) and converted to a 1-based inclusive interval [start+1, end], supporting both "chr" and non-"chr" naming conventions. At filtering time, each candidate variant, using its normalized position, is checked against the loaded intervals; any variant that falls inside a mask region is marked with the filter reason Masked. If no valid BED files are found, region masking is disabled and this is reported in the logs.

For each *de novo* candidate key that survives intersection and optional masking, HAT-FLEX performs a series of variant-level filters and annotations. Multi-allelic sites are counted for metrics but not discarded solely based on multiplicity. Symbolic or breakend-like normalized alternate alleles cause the site to be marked with the filter tag Symbolic. If the child's sex is unknown and the variant lies on the X chromosome or the Y chromosome, the site is marked SexSkip. Homopolymer filtering is applied and any site where either the reference allele or alternate allele contains an A or T homopolymer run of length greater than or equal to this threshold is marked Homopolymer. The user can specify a list of required FORMAT subfields with --require-format-keys (a comma-separated list, default GT,DP,AD,GQ), and if any required keys are missing for a variant, the site is marked MissingFORMAT. With --strict-format, such missing keys cause a fatal error; otherwise, these sites are skipped or marked while processing continues on.

For each candidate site, HAT-FLEX extracts the father, mother, and child sample fields based on the sample identifiers. If these fields cannot be retrieved due to malformed lines or similar issues, the site is marked SampleCols. Strand-bias filters are then applied if the SB FORMAT field is present, which is interpreted as four integers [ref_forward, ref_reverse, alt_forward, alt_reverse]. Two optional filters can

be activated: a minimum alternate allele read count on both strands and a Fisher's exact test on the 2×2 table of forward/reverse reads for reference and alternate alleles. If alternate allele reads on either strand are below the minimum, the site is flagged SB. If the right-tail Fisher p-value exceeds the specified maximum, the site is marked SBP. If no sb-fisher-pmax is provided, the Fisher test is skipped altogether. In allele-level intersection mode, if the specific alternate allele being considered is not actually present in the child's genotype at that site, the site is marked ChildNoAllele.

The pipeline then applies sex-aware haploid logic to determine which parent is considered relevant for each locus. At diploid loci (autosomes and PAR regions), both parents are relevant to filtering. At male non-PAR X chromosome loci, only the mother is considered relevant, and the father is effectively ignored; at male non-PAR Y chromosome loci, only the father is relevant and the mother is ignored. Irrelevant parents are recorded in the final VCF, and their fields can be masked as appropriate. Depth thresholds depend on both ploidy and the role of the sample. The default baseline minimum DP for diploid loci is 10, while the default for haploid loci is 5. The minimum DP for haploid loci is applied both to the child and to the relevant parent. Thus, each sample has an effective DP threshold based on whether the locus is haploid or diploid. The FORMAT fields DP (total depth), AD (allele depths), and GQ (genotype quality) are parsed if available; the alternate allele-specific read count is extracted from AD using the allele index. If this parsing fails, the site is marked BadFORMAT.

Evidence of alternate allele reads in the child are a requirement. At haploid loci (male non-PAR X chromosome and Y chromosome), the child must carry at least one alternate allele in the genotype, including partially missing haploid genotypes like ./1 or 1/.; otherwise, the site is marked ChildRef. At diploid loci, if the child's genotype is fully reference (i.e., contains no alternate alleles), the site is similarly marked ChildRef. Parent alternate allele-leak filtering is enforced using allele-depth-based thresholds and optional genotype-based checks. For each relevant parent, if the alternate read count exceeds the applicable maximum, the site is marked ParentAlt. A genotype-based leak check examines whether any relevant parent's genotype contains an alternate allele (including partially missing haploid genotypes such as ./1, 1/., . |1, or 1|.) and it triggers a ParentAlt flag, even if AD is missing or uninformative.

Depth and genotype quality filters are applied next. Sites missing any necessary DP or GQ values for the relevant parent(s) and child are marked MissingDPGQ. For each relevant parent, if DP is below the effective threshold, the site is tagged LowDP, and if GQ is less than or equal to some value (default 20), it is tagged LowGQ. The same checks are applied to the child using the appropriate haploid or diploid threshold. ALT read thresholds that produce ParentAlt are also enforced consistently at this stage. If both DP and AD are available for the child, the pipeline computes allele balance (AB) as alternate allele reads divided by total reads. At diploid loci, if AB is less than a value (default 0.25), the site is flagged LowAB. At haploid loci, an AB below a value (default 0.85) results in a LowAB_haploid tag. AB values are collected for downstream summary metrics.

HAT-FLEX also detects "haploid unusual" patterns at male non-PAR X/Y loci, where the child is haploid by expectation. A site is flagged as haploid unusual if the child's genotype is heterozygous (e.g., 0/1, 1/2) at a haploid locus or if the child's allele balance falls within a suspicious mid-range interval defined by a low value (default 0.35) and a high value (default 0.65), provided that partially missing haploid genotypes are not being skipped. The behavior for partially missing haploid genotypes (such as ./alternate allele or alternate allele/.) is controlled by arguments in the program. When skipping is enabled, these partial genotypes do not trigger the mid-range AB "unusual" flag, although heterozygous genotypes at haploid loci always do. Sites flagged as haploid unusual receive an INFO flag HAPUNUSUAL and a FILTER value HaploidUnusual.

In the output VCF, the pipeline preserves meta-information lines from the input and adds a ##HAT-FLEX_version=1.0 line if needed. It ensures that INFO fields required by the pipeline are present, including DENOVO (a flag marking *de novo* candidates), SEXCTX (a string describing sex/genomic context such as Autosomal, PAR, MaleX, FemaleX, or MaleY), IRRELPARENT (a string indicating which parent was ignored at haploid loci, F or M), HAPUNUSUAL (flag), and CLUSTER (a string, yes/no, indicating PASS-only clustering). It also defines a FILTER entry named HaploidUnusual to describe sites where a haploid locus shows a heterozygous-like genotype or suspicious mid-range AB. In the final header, sample columns are reordered so that only the trio samples are present, in the order father, mother, child. For each candidate site, the CHROM, POS, REF, and ALT fields are replaced by normalized coordinates (npos, nref, nalt), while other columns such as ID, QUAL, existing INFO, and FORMAT are preserved and updated with HAT-FLEX annotations. The SEXCTX INFO value is assigned as PAR for positions within pseudoautosomal regions, FemaleX for X chromosome variants in female children outside PAR, MaleX for non-PAR X chromosome variants in males, MaleY for non-PAR Y chromosome variants in males, and Autosomal otherwise. If exactly one parent is considered irrelevant in the haploid context, IRRELPARENT is set to F or M. The DENOVO flag is set on all candidate *de novo* sites that survive initial tagging, and irrelevant parents have their sample FORMAT fields replaced with a "." in the output VCF, ensuring downstream tools do not misinterpret their data at haploid loci where they are not informative.

For high-confidence PASS DNVs, HAT-FLEX annotates simple spatial clustering. For each chromosome, it maintains a sliding window of PASS variants using a deque of (position, record_index). The cluster window size is controlled set at a default of 100 bp (but can be varied) and the minimum number of PASS variants needed to define a cluster is set to 2 (but can be varied). When processing a PASS site, the pipeline first removes any previously stored PASS sites from the deque whose positions are more than cluster-window-bp bases away. If at least cluster-min-count - 1 PASS sites remain in the deque, the current site is considered part of a cluster. In that case, CLUSTER=yes is assigned to the current site and retroactively to all PASS sites still in the deque that previously had CLUSTER=no or no cluster annotation. PASS sites that do not meet the cluster criteria are annotated with CLUSTER=no. Variants that do not pass filtering never receive a CLUSTER annotation and do not contribute to the cluster window.

Each variant accumulates zero or more filter reasons, such as Masked, ParentAlt, LowDP, LowGQ, SB, LowAB_haploid, or HaploidUnusual. If no reasons are present, the FILTER field for the variant is set to PASS; otherwise, it is set to a semicolon-separated list of unique filter tags. By default, only PASS sites are written to the final VCF, but if --keep-failures is specified, both PASS and failing candidates are output and failures retain their filter tags in the FILTER column. Optionally, if a flag to output a table is provided, HAT-FLEX also writes a per-child TSV file (<child>.sites.tsv) with one row per variant that appears in the final VCF (PASS-only or including failures, depending on --keep-failures). This TSV includes the columns CHROM, POS, REF, ALT, final genotypes (F_GT, M_GT, C_GT), depths (F_DP, M_DP, C_DP), genotype qualities (F_GQ, M_GQ, C_GQ), and child allele balance (C_AB). For haploid loci, values from the irrelevant parent are replaced with a "." to match the masking in the VCF.

After writing a temporary plaintext VCF (<child>.final.tmp.vcf), HAT-FLEX attempts to compress and index the file using pysam (4-6), performing bgzip (7) compression to <child>.final.vcf.gz and optional tabix (7) indexing (none,tbi,csi) (default none). If pysam is not available or compression/indexing fails, the pipeline falls back to gzip compression without indexing. The temporary VCF is removed after compression, and the path to the final .vcf.gz file is printed to stdout for integration into downstream workflows. If a metrics flag is supplied, HAT-FLEX collects per-child summary statistics and writes them either as line-delimited JSON (when the filename ends with .json) or as tab-separated text with a header line. Metrics include counts of total candidate *de novo* records scanned (scanned), sites initially labeled

de novo (denovo_tagged), candidates not in masked regions (region_kept), sites with all required FORMAT keys (format_ok), sites passing strand-bias filters (sb_ok), and final PASS variants (final). It also records median depths (DP_F, DP_M, DP_C), median genotype qualities (GQ_F, GQ_M, GQ_C), and median child allele balance (AB_C), as well as percentages of variants on chrX, chrY, in PAR regions, and multi-allelic sites. Additionally, parent_leak_sites counts PASS variants where any relevant parent has non-zero ALT reads in AD despite passing other filters, providing a measure of residual parental signal. For reproducibility, HAT-FLEX writes a JSON run manifest per child (<child>.run_manifest.json) containing a timestamp of the run, all CLI arguments (with paths converted to strings), resolved input file paths (callers, family file, PED, regions, PAR BED), and output paths (final VCF and TSV, if emitted). Overall, HAT-FLEX is implemented in Python3 using the standard library and optionally pysam for bgzip compression and tabix indexing, and it transparently handles gzipped inputs, BED files, and BEDs packaged in tar archives or directories. A --dry-run mode reports which trios and files would be processed and the expected outputs without actually scanning variants or writing files, and verbosity can be increased with -v/--verbose (repeatable) for detailed logging of all processing steps.

SNOW Detailed Methods

SNOW (**S**econd-pass de **N**ovo variant **O**ffspring **W**orkflow, https://github.com/tycheleturner/snow) takes DNVs from HAT-FLEX or other DNV callers and performs second-pass cleaning, characterization, and annotation of the DNVs. It consists of several subtools including consolidate, parentcheck, phaser, pangenomecheck, table2snow, snow2table, snow2bed, and snow2acorn. Each of these are described below.

snow consolidate: merging adjoining calls on the same allele

Snow consolidate post-processes DNV calls by collapsing nearby child-only variants into single records (when applicable). The code takes as input a VCF containing candidate DNVs, a BAM/CRAM file for the child, and the corresponding indexed reference genome fasta file, along with the child's sample ID. For each contig, variants are traversed in genomic order. A *de novo*-like configuration is enforced at the genotype level: the child must carry at least one alternate allele, whereas the parents must be homozygous reference. In addition, only variants that either have no FILTER entries or are explicitly marked PASS and that carry a configurable cluster flag in the INFO field (default CLUSTER) are eligible for merging by *snow consolidate*. The CLUSTER tag is generated for all DNVs called with HAT-FLEX. For researchers who have DNVs from other programs, the CLUSTER tag can be generated by running *snow table2snow*. Variants satisfying these criteria are grouped into clusters when they occur within a user-defined window. For each cluster containing more than one variant, the code constructs a merged variant record by defining the minimal genomic interval spanning all clustered sites, fetching the corresponding reference haplotype from the reference genome fasta, and then building the alternate haplotype by walking along this interval and replacing each reference segment with the alternate (ALT) allele for that site (while disallowing overlapping variants). It than checks the read data for presence of the variants. The resulting output is written as a new VCF record with reference (REF) equal to the reference haplotype and ALT equal to the assembled alternate haplotype, and is assigned a deterministic ID of the form CHROM_POS_REF_ALT. The QUAL of the merged record is taken as the maximum QUAL among the constituent variants, and the FILTER field is set to the union of all non-PASS filters present in the cluster (or PASS if none). INFO annotations are copied from the first variant in the cluster and augmented with MNV_MERGED (flag) and MNV_NVAR (number of merged sites); FORMAT/sample fields are copied from the same representative record. Variants that are not merged are written back with their original FILTER status, but only if they remain PASS. Prior to writing, the code also ensures that any FILTER IDs observed in the input VCF are defined in the header and adds definitions for common filters where needed. The final output VCF is bgzip-compressed and optionally tabix-indexed, producing a header-consistent, MNV-aware callset suitable for downstream analyses.

snow parentcheck: re-checking read support

Snow parentcheck interrogates parental and child sequencing data for evidence of the alternate allele at candidate DNVs. The program takes as input a VCF/BCF containing variant calls for the child, along with BAM/CRAM alignment files for the father and mother and a reference FASTA (for CRAM decoding). For each variant annotated with the DENOVO flag, the tool scans the corresponding parental alignments in a small window around the variant position and counts the number of reads that support the first ALT allele. Read support is evaluated at the haplotype level using aligned query-reference base pairs, with explicit handling of SNVs, multi-nucleotide variants (MNVs), and simple left-normalized insertions and deletions. Reads are required to anchor the reference position and, for indels, to show an exact match to the expected inserted or deleted sequence in the CIGAR-aligned pairs. For variants annotated with SEXCTX=PAR, the tool additionally queries the homologous pseudoautosomal region (PAR) on the partner sex chromosome (X↔Y) to capture shared coverage. The resulting counts of ALT-supporting reads in the father and mother are written back to the VCF as F_ALT_READS and M_ALT_READS INFO fields, respectively. Also, included is an INFO field called K_ALT_READS, which is the minimum reads containing the ALT variant in the child to be retained as a high confidence site (default=2). As a final cleanup step, pre-existing low-quality flags (LowAB_haploid and LowDP) are removed from the INFO field, and the output VCF is bgzip-compressed and tabix-indexed (unless indexing is explicitly disabled), producing a ready-to-use annotated variant file for downstream *de novo* validation and filtering.

snow phaser: parent-of-origin inference of de novo variants

Snow phaser infers the parent-of-origin (POO) chromosome of DNVs in a trio using a genome-wide trio VCF, an alignment file for the proband, and a VCF of DNVs in the child. This method incorporates both physical phasing and phase-by-transmission (PBT) information.

Snow phaser identifies PBT sites by scanning the full trio VCF in the windows, containing the DNV, specified by the user (default = 40,000 bp). If the VCF is not bgzip-compressed and tabix-indexed the PBT site search defaults to a genome-wide search. Only biallelic SNVs (single-base REF and ALT alleles) are considered as candidates. For each site, the code retrieves genotypes for the child, father, and mother and apply several filters: 1) sex chromosome handling: in male probands, non-PAR regions of the X and Y chromosomes are excluded from PBT discovery, since the child is effectively haploid at these loci; 2) genotype quality (GQ): The site is discarded if any member of the trio has genotype quality below a user-defined threshold (default is 20); and 3) child genotype: the code requires the child to be heterozygous with a genotype exactly containing one reference allele and one alternate allele.

For each DNV, the code first collects all child reads that support the DNV alternate allele. Reads are obtained from the child BAM/CRAM file in a window spanning the variant and its local context. Alternate allele support is determined in an indel-aware manner: 1) SNVs: the code requires that the base aligned at the variant position matches the ALT allele; 2) insertions: for REF length = 1 and ALT length > 1, the code requires that the read carries the appropriate inserted sequence following the reference anchor base, using the CIGAR mapping (aligned pairs) to distinguish inserted bases from reference-matching positions; 3) deletions: For REF length > 1 and ALT length = 1, the code requires a pattern of reference-only positions (no query bases) spanning the deleted block in the alignment, consistent with the expected deletion length. Only reads that unambiguously support the ALT allele are retained and used for phasing. For each PBT site in this specified DNV window, the code fetches child reads overlapping the SNV position and restricts to the subset of reads (or read pairs) that also carry the DNV ALT allele (i.e., the shared haplotype). For each such read, the code queries the base aligned at the PBT SNV: 1) if the read carries the PBT ALT allele, the read is assigned to the parent that is known (by PBT) to carry the ALT allele at that site; 2) if the read carries the PBT REF allele, the read is assigned to the opposite parent (since the other parental haplotype is implied); and 3) if the read carries neither REF allele nor ALT allele at that position, or the base is missing (e.g., due to an indel), it is ignored. For each PBT site, the code

counts the number of DNV-ALT allele reads supporting the paternal and maternal haplotypes. These counts are summed across all informative PBT sites within the window to obtain overall paternal support and maternal support for the DNV. A minimum number of informative reads can be enforced (default 1); DNVs with fewer supporting reads are reported with unknown parent-of-origin.
For male probands, non-PAR regions of the X and Y chromosomes are treated differently. Since these regions are hemizygous, parent-of-origin is determined without using PBT sites: any DNV on the non-PAR X is assigned maternal origin (inherited from the mother's X), and any DNV on the non-PAR Y is assigned paternal origin. This deterministic assignment is used for both SNVs and indels, with a nominal maximum confidence.

Statistical confidence estimation: When both paternal support and maternal support are available and their sum meets the minimum evidence requirement, the code infers parent-of-origin by majority support. If paternal_support > maternal_support, the code assigns POO = paternal; if maternal_support > paternal_support, the code assigns POO = maternal; ties default to "unknown". To quantify confidence, the code models the number of reads supporting the more-favored parent as a binomial random variable under a null hypothesis of equal parental contribution (p = 0.5). Let

$$n = n_{pat} + n_{mat}, \qquad k = \max(n_{pat}, n_{mat}).$$

the code tests whether the observed imbalance can be explained by random segregation of reads between the two parental haplotypes. Under the null hypothesis

$$H_0 : P(read\ is\ paternal) = P(read\ is\ maternal) = 0.5,$$

the paternal read count X follows a binomial distribution

$$X \sim Binomial(n, p = 0.5).$$

the code computes a one-sided tail probability for the more supported parent,

$$p_{val} = P(X \geq k) = \sum_{i=k}^{n} \binom{n}{i} (0.5)^n.$$

Since n can be moderately large, the code evaluates this sum in $\log_{10}$ space using

$$log_{10}\binom{n}{i} = [ln\,\Gamma(\mathrm{n}+1) - \ln\Gamma(\mathrm{i}+1) - \ln\Gamma(\mathrm{n}-\mathrm{i}+1)] \cdot \frac{1}{ln10},$$

and

$$log_{10}P(X = i) = log_{10}\binom{n}{i} + ilog_{10}(0.5) + (n-i)log_{10}(0.5),$$

followed by numerically stable log-space summation.

From the p-value the code defines a simple [0,1] "phasing confidence" as

$$CONF = 1 - p_{val},$$

with values truncated to the interval [0,1]. The code also computes a Phred-scaled confidence score:

$$Q_{POO} = -10log_{10}(p_{val}),$$

which the code rounds to the nearest integer and caps at 255. Extremely small p-values ($\log_{10} p_{\mathrm{val}} \leq -300$) are treated as $p_{\mathrm{val}} \approx 0$, yielding CONF $= 1$ and $Q_{\mathrm{POO}} = 255$. When there are no informative reads ($n = 0$) or a tie ($n_{\mathrm{pat}} = n_{\mathrm{mat}}$), the code reports CONF $= 0$ and $Q_{\mathrm{POO}} = 0$. For male children, DNVs on the non-PAR X and Y chromosomes are assigned parent-of-origin deterministically based on hemizygosity (X → maternal, Y → paternal). In these cases, the code sets CONF $= 1$ and $Q_{\mathrm{POO}} = 255$ independent of read counts, as the parent-of-origin is implied by chromosome content rather than read-level phasing.
Several metrics are output from *snow phaser* in a VCF and/or table. These include POO_PAT_SUPPORT: paternal supporting read count, POO_MAT_SUPPORT: maternal supporting read count, POO_N_PBT: number of informative PBT SNPs used, POO_CONF: confidence score (1 − p-value), and POO_PHRED: Phred-scaled confidence.

snow pangenomecheck: pangenome graph membership annotation

Snow pangenomecheck annotates DNVs in an input VCF according to their presence in a pangenome graph represented as a bgzipped, tabix-indexed graph VCF. The method is designed for settings in which the reference graph VCF is large, but the query DNV VCF contains relatively few variants, making per-variant regional lookup more efficient than full-file joins. The workflow takes three required inputs: an input VCF, an output VCF, and a graph VCF. Optional arguments allow the user to define custom INFO field names for site-level membership, allele-level membership, and graph-derived allele frequency; to normalize chromosome naming conventions between files; and to suppress creation of an output tabix index. The tool uses pysam for VCF parsing and indexed querying, with warnings from CRAM multiple-iterator support suppressed to reduce log noise. For each record in the input VCF, the algorithm extracts chromosome, position, reference allele, and alternate alleles. It then queries the graph VCF over a narrow genomic interval spanning the record position using tabix-backed random access. When chromosome normalization is enabled, both chrN and N (where N is the chromosome) naming conventions are queried to accommodate mismatched contig styles across files. Site-level presence is defined as the existence of any graph VCF record at the same genomic position. Allele-level presence is evaluated more stringently by requiring an exact match on chromosome, position, reference allele, and alternate allele. For records without alternate alleles, only site-level membership is assessed. For records with one or more alternate alleles, the method initializes a binary flag for each ALT allele and scans all graph VCF records returned for the queried interval. Records are ignored unless they match the query position exactly; when chromosome normalization is enabled, normalized contig names must also agree. Exact allele matching is then performed by comparing the input reference allele and each alternate allele against the graph VCF alleles. Symbolic alleles and missing ALT values are skipped. Each ALT allele is annotated with a value of 1 if at least one exact graph match is found and 0 otherwise. In addition to membership, the method collects graph-derived allele frequencies from the graph VCF INFO/AF field. For each matched ALT allele, the corresponding AF value is extracted and stored. If multiple graph records contain the same ALT allele, the maximum AF across matching records is retained. ALT alleles with no match receive a default AF of 0.0. Thus, each input record is annotated with three outputs: a site-level indicator, an allele-level vector of membership flags, and an allele-level vector of pangenome allele frequencies. Before writing output, the program ensures that the required INFO header definitions are present in the VCF header. These include a scalar integer field for site-level graph membership, a per-allele integer field for ALT-level graph membership, and a per-allele float field for graph allele frequency. Annotated records are then written to the output VCF sequentially. After all records have been processed, the output VCF is compressed with bgzip-compatible indexing through pysam when available; otherwise, standard gzip compression is used as a fallback. Optional tabix index creation can be disabled. The program also reports summary statistics in verbose mode, including the number of records processed, the total number of ALT alleles encountered, the number of sites detected in the graph, and the number of ALT alleles matched in the graph. Overall, this method provides a lightweight per-record annotation strategy for determining whether input VCF variants are represented in a pangenome graph, while also propagating graph allele-frequency information for matched alternate alleles.

snow table2snow

Snow table2snow transforms a simple DNV table into a trio-based, bgzipped VCF that is ready for downstream processing in the SNOW pipeline. The script accepts a tab-delimited input table with four required columns (chrom, pos, ref, alt, with an optional header) and a bedtools-style (8) genome file that defines contig lengths and ordering. Variants are parsed, combined with the contig order from the genome file (with any extra contigs from the table appended), and sorted using a genome-driven chromosomal ordering scheme. For each variant, the tool constructs a VCF record with fixed, synthetic genotypes (father 0/0, mother 0/0, child 0/1), encodes trio relationships via a ##PEDIGREE header line, and tags all sites as *de novo* using an INFO flag DENOVO. Sex context is annotated in SEXCTX based on the child's sex (if

provided) and PAR intervals: if --child-sex is omitted, X and Y chromosome variants are excluded and all remaining sites are labeled as autosomal; otherwise, PAR coordinates are taken from an optional BED file or from built-in hg38 defaults, and each site is classified as Autosomal, PAR, MaleX, FemaleX, or MaleY. In addition, the script computes a PASS-only clustering annotation (CLUSTER) using a sliding window over positions within each chromosome: variants falling within a user-defined base-pair window and meeting a minimum count threshold are marked CLUSTER=yes, and earlier variants in the same window are retroactively upgraded to CLUSTER=yes as well, mirroring the clustering logic in HAT-FLEX. The final VCF includes standard ##contig and INFO/FORMAT header lines, is written as an uncompressed temporary VCF, then bgzip-compressed and optionally tabix-indexed (TBI or CSI) using pysam, with a gzip-only fallback if indexing is unavailable. The path to the final .vcf.gz file is reported on stdout for seamless integration into automated workflows.

snow snow2table

Snow snow2table converts a SNOW VCF or gzip-compressed VCF into a tab-delimited table while preserving the core variant fields and expanding INFO annotations into separate columns. The method is intended to facilitate downstream inspection and analysis of VCF content in tabular workflows. The program accepts an input VCF, an output table path, and an optional string used to represent missing values. Both input and output may be either uncompressed text files or gzip-compressed files, with file handling determined automatically from the filename suffix. The conversion is performed in two passes over the input file. In the first pass, the program scans the VCF to determine the set of INFO keys present across all records and to identify sample names from the #CHROM header line, if available. Metadata lines beginning with ## are ignored. When the column header line is present, sample names are extracted from columns following the standard fixed VCF fields. If no VCF header is detected but genotype/sample columns are present in variant records, the method infers the maximum number of sample columns observed and assigns placeholder names (SAMPLE1, SAMPLE2, etc.) to preserve those fields in the output. During this scan, the INFO field of each variant is parsed to collect all observed INFO keys. Both key-value annotations and flag-style annotations are recognized, and the final INFO column set is sorted lexicographically to ensure a stable output schema. In the second pass, the program writes the tabular output. The output table begins with the fixed VCF-derived columns CHROM, POS, ID, REF, ALT, QUAL, FILTER, INFO, and FORMAT. These are followed by one column per sample and then one column per discovered INFO key, named using the prefix “INFO_”. Thus, each row contains the original core VCF content, raw per-sample entries, and a column-expanded representation of INFO annotations. For each non-header variant record, the method parses the standard VCF fields and fills missing or empty entries with defaults consistent with VCF conventions or user-specified placeholders. Specifically, empty IDs and QUAL values are replaced with “.”, and an empty FILTER field is replaced with PASS. The raw INFO string and FORMAT string are retained directly in the corresponding fixed columns or replaced with “.” when absent. Sample columns are populated in order from the variant record; if fewer sample values are present than expected from the scanned schema, the remaining sample columns are filled with the user-defined missing-value string. The INFO field is then parsed into a dictionary by splitting semicolon-delimited items. Entries of the form key=value are stored using the provided value, whereas flag-style annotations lacking an explicit value are encoded as True. For each INFO key discovered during the initial scan, the corresponding output column is populated with the parsed value if present in the current record, or with the missing-value string otherwise. This produces a rectangular table in which all observed INFO fields across the file are represented consistently, regardless of whether a given record contains a particular annotation. If no INFO keys are detected during the initial scan, the program emits a warning to standard error but still writes the table with only the fixed and sample columns. The overall result is a tab-delimited representation of the VCF that preserves the original variant records, expands INFO annotations into separate columns, and supports both compressed and uncompressed inputs and outputs.

snow snow2bed

Snow snow2bed converts a SNOW VCF or compressed VCF (.vcf.gz) into a 4-column BED file. The output BED contains the fields chrom, start, end, and name, where the name field is constructed as chrom_pos_ref_alt_sample. The method is intended to generate interval-based representations of sequence-resolved variants while preserving allele identity and sample provenance in the BED record name. The program accepts three required arguments: the input VCF, the output BED file, and a sample name. Input and output may be provided either as uncompressed text files or gzip-compressed files, with compression handled automatically based on the filename suffix. The sample name is used only to label BED entries and is not used to subset genotypes or filter records. Before conversion, the method scans the VCF header to identify sample names from the #CHROM header line. If no samples are present, a warning is emitted. If samples are present but the requested sample name is absent, the program terminates with an error. This validation step ensures consistency between the BED name field and the declared VCF sample metadata. The converter then iterates through all non-header variant records in the VCF. Records with fewer than five tab-delimited columns are skipped with a warning. For each valid record, chromosome, position, reference allele, and alternate allele field are extracted. The VCF position, which is 1-based, is transformed to a 0-based BED start coordinate by subtracting one from the reported position. If multiple alternate alleles are present in the VCF ALT field, they are split on commas and processed independently, producing one BED row per ALT allele. Records lacking usable ALT alleles are skipped. For each REF/ALT pair, the code first restricts processing to sequence-resolved alleles composed exclusively of the canonical DNA bases A, T, G, and C. Alleles containing other characters, including symbolic or non-sequence representations, are excluded with a warning. The BED end coordinate is derived from the allele lengths using simple variant-class logic. Single-nucleotide variants are represented as one-base intervals. Insertions, defined as ALT longer than REF with a single-base REF anchor, are also represented as one-base intervals at the anchor position. Deletions and other REF-longer-than-ALT events are represented by spanning the full reference allele length. Multi-nucleotide substitutions of equal REF and ALT length are represented across the full substituted interval. More generally, length-decreasing complex variants are treated as deletion-like and length-increasing complex variants as insertion-like. If interval derivation fails for any reason, the allele is skipped with a warning. For each retained allele, the method writes a BED record consisting of chromosome, 0-based start, computed end, and a composite name string formatted as chrom_pos_ref_alt_sample. This naming convention preserves the original VCF position and allele identity while associating the interval with the declared sample. In summary, this method converts sequence-resolved VCF variants into BED intervals on a per-allele basis, validates the requested sample name against the VCF header, skips malformed or non-DNA alleles, and encodes variant and sample identity in the BED name field.

snow snow2acorn: acorn-ready files

Snow snow2acorn converts a SNOW VCF or compressed VCF (.vcf.gz) into a tab-delimited table suitable for downstream statistical or visualization workflows in acorn (9). The script accepts a VCF (optionally gzip-compressed) and a user-specified child sample name (taken from the #CHROM header) and produces a TSV (optionally gzip-compressed) with one row per variant. In a first pass over the VCF, the program identifies sample names and enumerates all INFO field keys present in the body of the file. In the second pass, it skips all header lines and, for each variant record, extracts core positional and allelic information (chromosome, 1-based genomic position, reference allele, alternate allele), fixed VCF attributes (ID, QUAL, FILTER, raw INFO string, FORMAT string), and the full FORMAT field string for the chosen child sample, which is reported in a genotype column without further parsing. All discovered INFO keys are expanded into dedicated columns prefixed with “INFO_”, and their values are populated by parsing the semicolon-delimited INFO field (with flag-style entries encoded as “True” and missing entries filled with a user-defined placeholder, default “.”). The resulting table has a leading sample column, set to the

requested child name for every row, followed by the genomic and VCF-derived fields, and then the "INFO_*" columns, providing a flat, analysis-ready representation of the variant data.

Probit Analyses Detailed Method

Child-level affection status was analyzed using probit regression under a latent-liability framework. In this formulation, affected versus unaffected status is treated as an observed binary outcome arising from an underlying continuous, unobserved liability. The probit link maps the probability of affected status onto the standard normal scale, permitting interpretation of regression coefficients as shifts in modeled liability for affected status. Positive coefficients indicate higher modeled liability, whereas negative coefficients indicate lower modeled liability. This modeling strategy was motivated by the liability-threshold framework for sex-biased traits, in which the less frequently affected sex is conceptualized as having a higher threshold for affection on the latent liability scale. For conceptual grounding, male and female prevalence parameters of 0.05 and 0.01429 were used, corresponding to liability thresholds of $T_{male} = 1.645$ and $T_{female} = 2.189$. These values were used only to contextualize the latent-threshold interpretation and were not estimated from the analytic sample.

The probit analysis used a variant-level DNV table together with a denovolyzeR-based gene-level mutation probability table. Variant-level observations were collapsed to the child level after stratification by mutation class and genomic region. Variants were grouped into broad functional classes intended to reflect predicted biological severity. The principal damaging classes were loss-of-function (LOF) and missense variants, while synonymous variants were retained as negative-control classes. Each variant was also assigned to a genomic region category, including autosomal and X non-PAR regions for the primary burden analyses. Observed burden was calculated by summing, within each child, the number of variants in each mutation class and genomic region. Core burden variables included autosomal LOF, autosomal missense, X non-PAR LOF, and X non-PAR missense DNV counts. Synonymous burden variables were also defined for autosomal and X non-PAR regions as negative-control terms. Expected burden was derived from denovolyzeR-based gene-level mutation probabilities and summarized in sex-specific expected profiles. Autosomal expected burden was identical in males and females, whereas X non-PAR expected burden was adjusted using a male:female mutation-rate ratio parameter, $r = \mu_{male}/\mu_{female} = 3.0$. Under this parameterization, male X non-PAR expectations were reduced relative to female expectations to reflect sex-specific mutation-rate and transmission assumptions. Primary explanatory variables were defined as observed-minus-expected burden for autosomal LOF, autosomal missense, X non-PAR LOF, and X non-PAR missense classes. Synonymous burden terms were likewise represented as excess burden variables for autosomal and X non-PAR regions.

The primary additive probit model included autosomal and X non-PAR damaging burden terms, autosomal and X non-PAR synonymous burden terms, and sex:

$$\Phi^{-1}\{\Pr(Y_i = 1)\} = \beta_0 + \beta_{A,LOF} B_i^{A,LOF} + \beta_{A,MIS} B_i^{A,MIS} + \beta_{X,LOF} B_i^{X,LOF} + \beta_{X,MIS} B_i^{X,MIS} + \gamma_A S_i^A + \gamma_X S_i^X + \beta_F F_i.$$

Here, Y_i denotes affected status for child i, F_i is the female indicator, $B_i^{A,LOF}$ and $B_i^{A,MIS}$ denote autosomal excess damaging burdens, $B_i^{X,LOF}$ and $B_i^{X,MIS}$ denote X non-PAR excess damaging burdens, and S_i^A and S_i^X denote autosomal and X non-PAR excess synonymous burdens. Under this parameterization, damaging-burden coefficients represent shifts in modeled liability associated with higher excess burden in the corresponding class, the synonymous coefficients represent analogous shifts for negative-control burden terms, and the sex coefficient represents a shift in baseline liability on the probit scale after adjustment for the included burden variables.

To assess whether damaging-burden effects differed between males and females, a sex-interaction probit model was also fit:

$$\Phi^{-1}\{\Pr(Y_i = 1)\} = \beta_0 + \beta_{A,LOF}B_i^{A,LOF} + \beta_{A,MIS}B_i^{A,MIS} + \beta_{X,LOF}B_i^{X,LOF} + \beta_{X,MIS}B_i^{X,MIS} + \beta_F F_i + \delta_{A,LOF}(B_i^{A,LOF}F_i) + \delta_{A,MIS}(B_i^{A,MIS}F_i) + \delta_{X,LOF}(B_i^{X,LOF}F_i) + \delta_{X,MIS}(B_i^{X,MIS}F_i).$$

In this model, the male effect for burden class k is given by β_k, and the female effect is given by $\beta_k + \delta_k$. Thus, the interaction terms quantify the extent to which the association between excess damaging burden and modeled liability differs in females relative to males.

REFERENCES

1. McKenna A, Hanna M, Banks E, Sivachenko A, Cibulskis K, Kernytsky A, et al. The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. Genome research. 2010;20(9):1297-303.
2. Poplin R, Chang PC, Alexander D, Schwartz S, Colthurst T, Ku A, et al. A universal SNP and small-indel variant caller using deep neural networks. Nature biotechnology. 2018;36(10):983-7.
3. Yun T, Li H, Chang P-C, Lin MF, Carroll A, McLean CY. Accurate, scalable cohort variant calls using DeepVariant and GLnexus. bioRxiv. 2020:2020.02.10.942086.
4. Bonfield JK, Marshall J, Danecek P, Li H, Ohan V, Whitwham A, et al. HTSlib: C library for reading/writing high-throughput sequencing data. Gigascience. 2021;10(2).
5. Danecek P, Bonfield JK, Liddle J, Marshall J, Ohan V, Pollard MO, et al. Twelve years of SAMtools and BCFtools. Gigascience. 2021;10(2).
6. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, et al. The Sequence Alignment/Map format and SAMtools. Bioinformatics (Oxford, England). 2009;25(16):2078-9.
7. Li H. Tabix: fast retrieval of sequence features from generic TAB-delimited files. Bioinformatics (Oxford, England). 2011;27(5):718-9.
8. Quinlan AR, Hall IM. BEDTools: a flexible suite of utilities for comparing genomic features. Bioinformatics (Oxford, England). 2010;26(6):841-2.
9. Turner TN. Acorn: an R package for de novo variant analysis. BMC bioinformatics. 2023;24(1):330.