RESEARCH

OVAS: an open-source variant analysis suite with trait-penetrance modelling

Monika Mozere[1†], Mehmet Tekman[1†], Jameela Kari[2], Detlef Bockenhauer[1], Robert Kleta[1*] and Horia Stanescu[1]

Abstract

Background: The advent of modern high-throughput genetics continually broadens the gap between the rising volume of sequencing data, and the tools required to process them. The need to pinpoint a small subset of functionally important variants has now shifted towards identifying the critical differences between normal variants and disease-causing ones. The ever-increasing reliance on cloud-based services for sequence analysis and the non-transparent methods they utilize has prompted the need for more in-situ services that can provide a safer and more accessible environment to process patient data, especially in circumstances where continuous internet usage is limited.

Results: To address these issues, we herein propose our standalone Open-source Variant Analysis Sequencing *(OVAS)* pipeline; consisting of three key stages of processing that pertain to the separate modes of annotation, filtering, and interpretation. Core annotation performs variant-mapping to gene-isoforms at the exon/intron level, append functional data pertaining the type of variant mutation, and determine hetero/homozygosity. Up to 12 filtering modules can be used in sequence ranging from single quality control to multi-file penetrance model specifics such as X-linked recessive or mosaicism. Depending on the type of interpretation required, additional annotation is performed to identify organ specificity through gene expression and protein domains. In the course of this paper we analysed an autosomal recessive case study. OVAS made effective use of the filtering modules to recapitulate the results of the study by identifying a single novel promoter mutation from exome-capture sequence input samples.

Conclusion: OVAS is an offline open-source modular-driven analysis environment, designed to annotate and extract useful variants from Variant Call Format (VCF) files, and process them under an inheritance context through a top-down filtering schema of swappable modules, run entirely off a live bootable medium and accessed locally through a web-browser.

Keywords: open source; variant analysis; disease model; mosaic; bootable; live environment

Background

The technological evolution of sequencing platforms has progressed rapidly since the completion of the Human Genome project via Sanger sequencing methods [1, 2]. Modern high-throughput sequencing (HTS) approaches post-Sanger era have superseded this standard, allowing for a greater number of variants to be sequenced across the whole genome by employing powerful mass fragmentation/amplification approaches upon a target sequence [3, 4].

The raw sequence FASTQ reads produced by these HTS platforms are aligned to a specific version of the NCBI reference sequence and collated into a Binary Alignment Map (BAM) where variants of interest can then be individually "called" to form a Variant Call Format (VCF) file of novel or known variants conforming to a specific variant database (dbSNP) [5, 6].

BAM and VCF data are orthogonally related, with the former storing horizontal stretches of FASTA sequence reads aligned unevenly on top of one another forming "pile ups", and the latter taking vertical cross-sections of these pileups at specific loci to form a variant call.

The VCF specification was designed for the 1000 Genomes project to produce a robust format that could house the many samples often sequenced under the same batch, but has since been adopted

*Correspondence: r.kleta@ucl.ac.uk
[1]Division of Medicine, University College London, London, NW3 2PF UK
Full list of author information is available at the end of the article
†Equal contributor

by projects such as UK10K, dbSNP, NHLBI Exome Project, amongst others. The format is flexible with annotations, where additional fields can be outlined in the header and adhered to in the body of the data. Each line of the VCF body describes a single variant; physical position paired with a reference allele (as ascribed by a reference genome consistent across the entire VCF file) and alternate alleles that appear within samples. Major and minor alleles are specific only to the sample population but their frequencies can be pre-computed and appended to a variant line as additional information to then be utilized in small population analyses such as inheritance modelling [6].

Variant analysis suites all work under the same principle; filtering all variants under a user-specified set of criteria against the various variant annotations present in the VCF in order to produce a subset informative to the phenotype. Optimistic filtering measures will produce a smaller set with the drawback of missing key causative variants, and conservative filtering measures will produce too many false positives. The effectiveness of an analysis rests primarily upon the accuracy of the variant annotations which can attribute to as much as 15% of false negatives [7], as well as the frequency of false negatives that are discarded due to overly-stringent quality filtering. A common approach to addressing both issues is through learning algorithms that can be trained to favour individual variants over others with the caveat of producing results via 'black-box' methods that may create some disparity between the user and their data [8].

A more transparent approach is to expand the scope of the filtering beyond the variant / gene-level and explore variants under a larger trait-penetrance context.

Mendelian traits conform to the four classical modes on inheritance of autosomal / X-linked, dominant / recessive penetrance. Dominant disorders result from the inheritance of a single mutant allele which is manifested in each subsequent generation with a 50% chance of likelihood in offspring from a single affected parent. Recessive traits require the inheritance of two mutant alleles on opposing strands in order to block any functioning copies of the causative gene. Parents are typically carriers with affected offspring. These disorders are at times a result of consanguineous marriages, where a single mutant allele manifests on both alleles due to the multiple paths of descent it can undertake [9]. In the case of X-linked recessive inheritance, males with a single mutant copy are hemizygous and must express the phenotype.

For non-Mendelian disorders, we also consider the special case of *mosaicism*; where de novo mutations produce two or more populations of cells that result in segregated sets of genotypes within the same individual. Mosaic genotypes can be revealed stochastically by measuring alternate allele frequencies against expected values [10].

Here we outline our Open-source Variant Analysis Suite (OVAS) that makes use of these inheritance modelling scenarios with the aim to vastly reduce the number of false positives.

Implementation

The core ideology behind OVAS was to preserve the VCF specification at each step of the analysis, and this is catered to extensively within the pipeline where each module inputs and outputs VCF file(s) in order to facilitate the chaining of subsequent pipeline modules downstream. This allows for full analysis transparency, where results can be extracted at any stage of an ongoing analysis.

Module ordering is flexible in this regard, with the exception of the primary annotation modules which are required to run prior to any filtering in order to produce an effective analysis of the variants. Pre-existing gene and function annotations within input data are ignored unless generated by a previous run of the OVAS pipeline, supplanting foreign annotations with the pipeline's own if required. This is to ensure unambiguous results stemming from external annotations using unknown sources that may result in erroneous output variants.

OVAS is rooted firmly in trusted public domain databases such as RefGene, dbSNP, UniProt, and many others accessed through the widely-used UCSC Genome Browser [11], ensuring a beneficial accordance between the variants described in both the Genome Browser and OVAS. The explicitly open nature of pipeline also prompts a predilection towards open-source or scripted languages and frameworks, which further serve to uphold the confidence between the end-user and their data.

Though core operations are managed primarily through back-end shell scripts, the pipeline can be accessed and configured through a web-front interface in order to cater for simplicity and user-operability. Users can upload their data either through the web-interface or by manual file placement as preference dictates.

OVAS consists of a series of inter-connecting Bash shell scripts which serve as necessary framework to accommodate wrappers for subsequent modules in order to chain (or "pipe") them together, as well as provide anchors for static and dynamic data management throughout general operation as shown in Figure 1.

0.1 Application Suite and Interface

The pipeline was originally developed in a headless Linux shell environment to be deployed on any Unix-like system that supports Bash, appealing to experienced technicians who can perform their own input

validation. However, significant effort was made to include researchers from non-computing backgrounds who could benefit from the rich processing without the cost additional groundwork.

0.1.1 Web-front

To necessitate the uptake of OVAS, a web interface was created to facilitate input validation and pipeline configuration process.

The file upload procedure is streamlined by means of a pedigree file which pre-specifies cases (affecteds) and controls (unaffecteds) as well as their relation to one another. Pedigree data is automatically parsed into a table of cases and controls for all detected families, with file upload slots being assigned for each individual. The user is then given the choice to select multiple VCF files which are then mapped programatically to the appropriate individual slots, or separate files can be dragged and dropped if more manual assignment is desired.

The interface extends to display configuration options for each annotation and filtering module whilst uploading occurs in the background. Modules are enabled by expanding check-boxes to display individual module parameters and thresholds that can be overridden by user criterion, examples of which can be shown in Figure 2.

A drop-down box of available penetrance model provides mutually-exclusive model-dependent options to better refine the analysis, such as parent or unaffected sibling-specific filtering. Additional annotation requirements are set (or skipped upon preference) and then the pipeline is run in tandem to the existing input session. In the case of user-termination, re-upload is not necessary for the same analysis as the process will reuse the temporary files from the last session and will not repeat the same work twice, resuming from where it left off. Once complete, the pipeline self-terminates and produces an interactive report of the remaining variants primed for feature presentation/concealment to help pinpoint variants of interest such as those shown in Figure 3.

The pipeline is spawned in a GNU *screen* session in order to enable process control and resumeablility, where snapshots of a session in-process are repeatedly retrieved from the shell process to the web front-end via *PHP* scripts. UI elements are managed with CSS and minimal JavaScript, with the exception of the interactive report which performs table operations primarily through the latter. The front-end itself is hosted via a minimal *lighttpd* server, and ongoing OVAS processes can be managed both from the web-interface as well as from the shell provided in the live environment.

0.1.2 Self-Contained Environment

The full OVAS suite comprises of the core pipeline processing back-end encapsulated by the web-interface to handle input validation, which is encapsulated once more by a live operating system that handles and provides general file utilities as well as overall startup. Each of these three components exist as separable peripherals, but are optimal in the above configuration by facilitating and abstracting the installation of each through the use of symbolic links and providing constant anchors for static data bundled with the environment.

Arch Linux was chosen as the environment backbone due to it being a lightweight "no-frills" operating system that does not come pre-packaged with desktop sessions (and their associated bloatware). OVAS runs straight off the X desktop server with the OVAS interface autostarted along with a minimal dock for spawning additional applications [12].

The static data primarily encompasses a variety of gene map configurations from human genome reference version hg18 through to hg38, as well as the raw nucleotide FASTA files for each chromosome specific to the versions, amounting to 15GB of genomic data. Due to the packing process, as well as compression algorithm used in the Squash Filesystem creation process [13], OVAS mounts up to no more 2.7GB. This makes it ideal for bootable mediums such as DVDs and USB sticks, where the latter can preserve data across subsequent sessions.

0.2 Pipeline Modules

Each module is tasked with the function of separating variants from an input file into two distinct output VCF groups of "filtered" and "discarded"; with the former group being passed into the next module, and the latter being halted at the current point of processing to be stored for potential debugging purposes. The discard process at each module lends a progressive performance increase in the processing speed of each subsequent module due to the input being only a subset of the input that came before it, whilst still retaining the aggregate total of discarded variants at each step.

0.2.1 Pre-processing

All VCF files immediately undergo initial preparation upon file upload from the web interface, where a background shell script renames the files to better emulate their pedigree counterparts, and asserts that all variants are in correct order following a chromosome:position sorting key.

0.2.2 Core Annotation
The annotation stages of the pipeline then prime the variants with relevant metadata that will then be filtered against user-criterion throughout the rest of the pipeline. The annotation stage is the only mandatory stage of the pipeline, and a great portion of filtering occurs at these stages too, with up to 90% of true negatives being discarded. As a result of the large demand placed upon the modules at this stage, they were written in the C++ in order to reduce time and memory constraints on low-end platforms. The stage is split into three modules (in order of processing):

Adding Genes
Appends a gene-context to the variants under a user-configured level of detail at the gene/intergenic junction or the exon/intron/splice/UTR sub-divisions, including isoforms. Regulatory variants further up or downstream of UTR can be specified by defining custom margins of enclosement, and wholly intergenic regions are discarded by default (though can be kept upon user preference).

Adding Function
Applies functional annotation upon the variants processed in the previous step; performing a cDNA lookup of where a variant falls within the coding portions of the gene in order to predict the type of mutation (missense, synonymous, or non-synonymous) at the codon and subsequent amino-acid level. Anti-sense encoded genes are handled accordingly, and for insertion/deletion (indels) variants the module performs the required addition/subtractions across a consistent reading frame to discern the mutation.

Adding Zygosity
Addresses a confidence issue in with pre-processed variants, where heterozygosity and homozygosity would be assigned based on post-quality filtering metrics. This module sets zygosity by nucleotide base-count alone, and determines HET/HOM assignments based upon a user-set frequency threshold (default < 0.65).

Once fully annotated, the resultant output VCFs are ready to be processed by the filtering modules.

0.2.3 Filtering Modules
The filtering modules consist of a series of Python (v2.7) scripts designed to parse these fields with the aim of minimizing the need for any mapping or additional pass-throughs. A variant line in a VCF file describes the eight mandatory fields grouped into three distinct categories (in order of filtering complexity):

Variant Properties
The first five mandatory columns (chromosome, physical position (base-pairs), variant-marker identifier, reference allele, and alternate alleles) are processed by two main modules: **Physical Location Filter**, which parses chromosome and physical position to keep variants inclusive to user-set loci ranges; and **Novel Variant Filter** which discards all variants with pre-existing identifiers (i.e., not ".").

Variant Metadata
Here, the variant call information field (INFO) is processed, which consists of variant-calling properties summarizing the FASTA strand pileup bisected by the variant. The INFO field alludes to the quality of the sample data, but not to the sample data itself, enabling for fast single-pass processing by the following four modules: **Read Depth Filter** and **Call Quality Filter**, which both discard variants falling below a user-set limit, the former upon the number of FASTA reads aligned at that position, and the latter upon the variant calling score value; **Mutation Type Filter**, which makes use of functional annotations in order to filter single variants based upon user-set requirements of including any (multiple) of missense, nonsense, and synonymous mutation types; and finally **Same Gene Filter** and **Same Variant Filter**, that act upon multiple VCFs and produce a common output set of variants that reside in either the same gene(s) or share identical variants respectively.

Sample Data
The sample format field (FORMAT) structures the presentation that all subsequent sample data conform to, and must be parsed before handling of the actual data. The **Alternate Allele Frequency** module scans the sample data in order ascertain the absolute frequencies of the alternate allele(s) in the population whilst removing variants with frequencies exceeding user-defined upper/lower-bound thresholds.

0.2.4 Inheritance Filtering
This section performs trait penetrance modelling for differently affected individuals following sibling-sibling, and sibling-parent relations. For all detected parent-offspring trios, variants undergo context-based filtering depending on the penetrance-model specified:

Autosomal Dominant
The phenotype is caused by a single mutant autosomal allele, and affected individuals must have affected parents, mapping any $\{\text{HOM,HET}\}\mapsto\{\text{HET,HOM}\}$ under complete penetrance. Under a *de novo* context

all common affected variants are filtered against unaffected controls, otherwise variant commonality is kept within sibling groups.

Autosomal Recessive
The phenotype is caused by a loss of function stemming from both copies of an autosomal gene, at times from the result of consanguineous breeding. Two paths of transmission are considered from parent↦offspring depending on whether the affected offspring variant is compound-heterozygous (C-HET) or homozygous (HOM):

1 **HOM**, At least one parent maps HOM↦HOM, or {HET/HET}↦HOM if both parents are carriers.
2 **C-HET**, Parents are assumed to be carriers for different singular HET variants across common genes, which compound in offspring as multiple HET variants within the same gene. Under a gene context, this is resolved via {HET1/HET2} ↦ {HET1+HET2} mapping to produce a C-HET gene.

Siblings are then filtered for common variants existing within affecteds siblings only, discarding those that are homozygous in unaffected controls.

X-linked Dominant
As with autosomal dominant but with the mutant allele on the X-chromosome.

X-linked Recessive
As with autosomal recessive but with mutations occurring on the X-chromosome. Males with a single mutant copy are hemizygous and are treated as homozygous, exempting them from compound heterozygosity checking.

Mosaicism
Mosaic inheritance is treated as a special case, where allele frequencies are pre-calculated for each variant and then filtered against user-set thresholds conforming to expected mosaic frequency ranges (typically between 10-35%).

0.2.5 Extended Annotation
The last stage of pipeline constitutes a small subset of variants which have successfully passed through the main filtering stages and require finer analysis which is enabled by providing an even greater context to compare the variants. Additional annotation relates to the downstream effects of said variants such as structure, function, and expression.

Isoform Context
Translates gene isoforms into their RefSeq nomenclature counterparts.

Protein Context
Assigns protein annotation information from UniProt sources to assign information related protein domain.

Gene Expression
Organ and tissue-specific data from the Encode GNF Atlas2 database is provided along with expression ratios which can be further filtered against user-specified limits.

Results

Case Study

Three families presented with hyperinsulinemic hypoglycemia and congenital polycystic kidney disease (HIPKD), a rare heterogeneous pair of disorders following an autosomal recessive disease pattern. Exome-capture sequencing and whole-genome linkage analysis revealed a promoter mutation within a significant locus on chromosome 16 [14].

Of the 9 input VCF files, 4 were affected cases of which 2 were siblings. The remaining VCFs were controls consisting of 4 (2 pairs) of unaffected parents and 1 was unaffected sibling, permitting the use of variant-level filtering.

Each VCF file comprised of approximately 250,000 variants (SNPs and InDels) and were profiled against a gene map at the first annotation step (*Adding Genes*) comprised of exons, introns, 5' and 3' UTR, and essential splice sites (5bp). Gene isoforms as well as regions deemed wholly intergenic were also retained.

Core annotation normally accounts for a vast majority of non-coding variants being filtered out, but due to the regulatory nature of the analysis all intergenic variants were kept, resulting in no implicit filtering occurring during this initial stage.

Prior linkage analysis hinted at a small locus of interest with a very significant LOD-score (>6) and this, in conjunction with the *Call Quality Filter* (≥20) resulted in 99.9% of the variants being filtered out in all cases, bringing the number of variants in each to less than 100 as shown in Figure 4. The rarity of the phenotype prompted a search for novel variants, resulting in under 10 potentially causative-variants.

Applying the autosomal recessive inheritance module with same variant filtering further reduced the the total number of variants, and a final commonality filter set at the variant level identified a single causative intergenic variant within the promoter region of PMM2 that was confirmed as the disease-causing variant.

Discussion

Depending upon the total input variants as well as the number and ordering of modules used, an average initial analysis using any number of modules (excluding alternate allele filtering) for VCF files containing 300,000 variants each, will attribute a total of 2 minutes per VCF.

There are several limiting steps however, with the largest bottleneck occurring at initial gene annotation stage, which must prime all input variants for downstream filtering through the use of a gene (or exon) map that is dependent upon user parameters. Gene maps for a variety of user parameters already exist as static files in the live environment, but not all use-cases are covered and a new gene map must be generated for custom configurations which can take up to 1 hour to retrieve depending on internet speed and proximity to the closest UCSC MySQL mirror.

In the case of general gene map use-cases, the *Adding Genes* annotation step still requires 200 times more processing time than most other modules, and was the sole reason that all annotation modules were re-written in C++ to benefit from a significant performance increase that reduced the module's processing time from an initial time of 10 minutes to under 3 minutes (Table 1).

The rest of the annotation modules are comparatively much faster, with the functional annotations experiencing mild latency related to disk read speeds when performing repeated byte-offset lookup upon FASTA files. The initial sorting of the variants upon file upload is valuable in this regard due to the higher tendency of adjacent variants to share the same disk cluster and reap paging benefits.

The last noticeable slowdown occurs within the Javascript-powered interactive report and is dependent upon the number of final variants it has to tabulate, where the difference between 1,000 and 10,000 final variants maps to a range of 1 to 15 seconds. Across subsequent pipeline runs, processing is not repeated for the same data; each module checks whether an input VCF file has already been processed by the current pipeline configuration, and repeatedly iterates through the module ordering until the last processed input set is reached where it can resume processing.

0.3 Transparency and Deployment

The portability of OVAS grants a significant advantage over present-day web-based pipelines by keeping all analyses securely *in situ*, which is greatly beneficial to regions of the world without consistent or active internet in addition to researchers handling personal or private data.

Cloud-based analyses require input data to be uploaded to an external server in order to perform processing, and data ownership after upload is not always retained especially in the case where the work was performed within the cloud [15]. Further, many cloud-services employ non-transparent proprietary methods to reduce the number of false-positives and false-negatives. A common approach is to make use of an internal database or learning algorithm that favours some variants over others based on previous analyses (or a similar training set) [8], resulting in informative variants produced by unquantifiable "black-box" means, creating disparity between the end-user and their analysis.

Transparent filtering methods are likelier to instil greater confidence in the data with the added benefit of customization to better tailor a filter to an analysis in the case of open-source implementations, as with the case of OVAS.

Conclusions

The self-contained environment provided by OVAS allows researchers to tailor all aspects of their analysis and retain control of their data sets at any phase of processing by means of the transparent open-source modules that comprise the pipeline.

The live environment, paired with the web front-end, provides the additional advantage of abstracting the end-user from the underlying platform specifics by streamlining the input and configuration process, as well as logging active progress descriptions for the current stage of processing, and lastly providing a malleable final report upon all remaining variants discovered complete with dynamic filtering capabilities. The entirety of all uploaded variants are processed first at the gene annotation stage, placing significant strain at the initial stage of the pipeline that is only managed through the use of employing C++ binaries to overcome the performance bottleneck that would otherwise exist with Python/Bash scripts.

The annotation step is crucial, especially for whole-genome sequence data where the vast majority of the variants would be deemed wholly intergenic and would be filtered out as uninformative to the analysis. More common exome-sequencing data typically observe less of a reduction at a much faster processing rate due to the smaller number of total variants, but at the impediment of missing regulatory elements due to lack of coverage. Modules downstream of the annotation stage run trivially, and due to the pipeline's resume feature which prevents OVAS from processing the same data twice, many subsequent analyses with different module configurations can be run in quick succession after the initial annotation step is complete.

OVAS is future-secure due to the inclusion of the background scripts that generated the static data being packaged with the live environment. Updates to the human genome reference, variant databases, and FASTA sequences can be retrieved on demand for platforms with active internet connections. Changes will preserve across successive boots for non-volatile storage mediums such as USB sticks, ideal in deployment scenarios with infrequent or absent internet access.

Availability of Data and Materials

OVAS was developed under C++, Python, and Bash shell scripts. It is free software licensed under GPLv3, with the source code and live ISO binary image freely accessible for download at https://www.bitbucket.io/momo13/ovas-pipeline.git. The data that support the findings of this study are available at the code repository and also within the live ISO environment. The OVAS pipeline can either be directly installed locally on a pre-existing Linux OS, or it can be accessed in-situ by booting the live image.

List of Abbreviations

UTR: Untranslateable Region, **cDNA**: Complementary DNA
HET: Heterozygous, **HOM**: Homozygous
C-HET: Compound-Heterozygous, **SNP**: Single Nucleotide Polymorphism
InDel: Insertion-Deletion, **LOD**: Logarithm of the Odds
HTS: High-Throughput Sequencing, **VCF**: Variant Call Format
BAM: Binary Alignment Map
HIPKD: Hyperinsulinemic hypoglycemia and congenital polycystic kidney disease

Ethics approval and consent to participate

Ethics approval and consent was provided by the ethics committee of Royal Free Hampstead NHS Trust (committee's reference number R&D ID 7727).

Competing interests

The authors declare that they have no competing interests.

Author's contributions

MM designed and implemented the filtering, extended annotation, and disease model-specific scenario Python modules. MT wrote the core annotation C++ utilities, and reworked the pipeline into the live ISO bootable environment. The pipeline workflow was structured by MM and implemented by MT. The study of genomic data sets given by JAK and DB prompted the conception of the pipeline. HS was instrumental to the development process by providing method evaluation, feature recommendation, and overall technical supervision. RK provided quality control assessment and agreed to be accountable for all aspects of the work in ensuring that questions related to the accuracy or integrity of any part of the work are appropriately investigated and resolved. All authors considered, discussed, read, and approved the final manuscript.

Funding

This work was supported by St. Peter's Trust for Kidney, Bladder and Prostate Research, the David and Elaine Potter Charitable Foundation, Kids Kidney Research, Garfield Weston Foundation, Kidney Research UK, the Lowe Syndrome Trust, the Mitchell Charitable Trust, and the European Union, FP7 (grant agreement 2012-305608 "European Consortium for High-Throughput Research in Rare Kidney Diseases (EURenOmics)"). Part of this work was also supported by the Deanship of Scientific Research, King Abdulaziz University, Jeddah, grant number 432/003/d to JAK, DB and RK.

Author details

[1]Division of Medicine, University College London, London, NW3 2PF UK.
[2]Pediatric Nephrology Center of Excellence and Pediatric Department, Faculty of Medicine, King Abdulaziz University, Jeddah, Kingdom of Saudi Arabia.

References

1. Lander ES, Linton LM, Birren B, Nusbaum C, Zody MC, Baldwin J, et al. Initial sequencing and analysis of the human genome. Nature. 2001;409(6822):860–921.
2. Sanger F, Nicklen S, Coulson AR. DNA sequencing with chain-terminating inhibitors. Proceedings of the National Academy of Sciences. 1977;74(12):5463–5467.
3. Lengauer T. Bioinformatics-From Genomes to Therapies. Wiley Online Library; 2007.
4. Bockenhauer D, Medlar AJ, Ashton E, Kleta R, Lench N. Genetic testing in renal disease. Pediatric Nephrology. 2012;27(6):873–883.
5. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, et al. The sequence alignment/map format and SAMtools. Bioinformatics. 2009;25(16):2078–2079.
6. Danecek P, Auton A, Abecasis G, Albers CA, Banks E, DePristo MA, et al. The variant call format and VCFtools. Bioinformatics. 2011;27(15):2156–2158.
7. Warden CD, Adamson AW, Neuhausen SL, Wu X. Detailed comparison of two popular variant calling packages for exome and targeted exon studies. PeerJ. 2014;2:e600.
8. Pabinger S, Dander A, Fischer M, Snajder R, Sperk M, Efremova M, et al. A survey of tools for variant analysis of next-generation genome sequencing data. Briefings in bioinformatics. 2014;15(2):256–278.
9. Kari JA, Bockenhauer D, Stanescu H, Gari M, Kleta R, Singh AK. Consanguinity in Saudi Arabia: a unique opportunity for pediatric kidney research. American Journal of Kidney Diseases. 2014;63(2):304–310.
10. Biesecker LG, Spinner NB. A genomic view of mosaicism and human disease. Nature Reviews Genetics. 2013;14(5):307–320.
11. Karolchik D, Baertsch R, Diekhans M, Furey TS, Hinrichs A, Lu Y, et al. The UCSC genome browser database. Nucleic acids research. 2003;31(1):51–54.
12. Scheifler RW, Gettys J. The X window system. ACM Transactions on Graphics (TOG). 1986;5(2):79–109.
13. Lougher P, Lougher R. SquashFS; 2008. http://www.squashfs-lzma.org/.
14. Cabezas OR, Flanagan SE, Stanescu H, García-Martínez E, Caswell R, Lango-Allen H, et al. Polycystic Kidney Disease with Hyperinsulinemic Hypoglycemia Caused by a Promoter Mutation in Phosphomannomutase 2. Journal of the American Society of Nephrology. 2017;p. ASN–2016121312.
15. Reed C. Information 'Ownership' in the Cloud. Queen Mary School of Law Legal Studies Research Paper. 2010;(45).

Tables

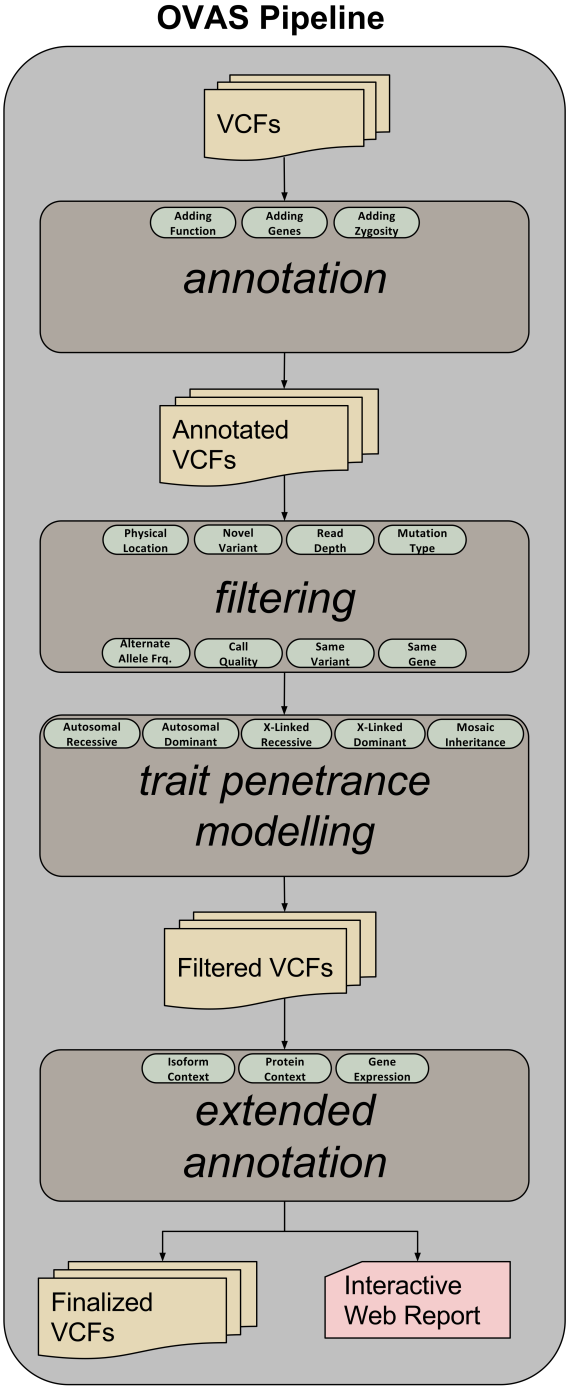

Figure 1 Overall structure of the OVAS pipeline: VCF files as referenced by a pedigree file are fed into the pipeline and are processed in turn by the core annotation, optional filtering, trait penetrance modelling, and additional annotation modules.

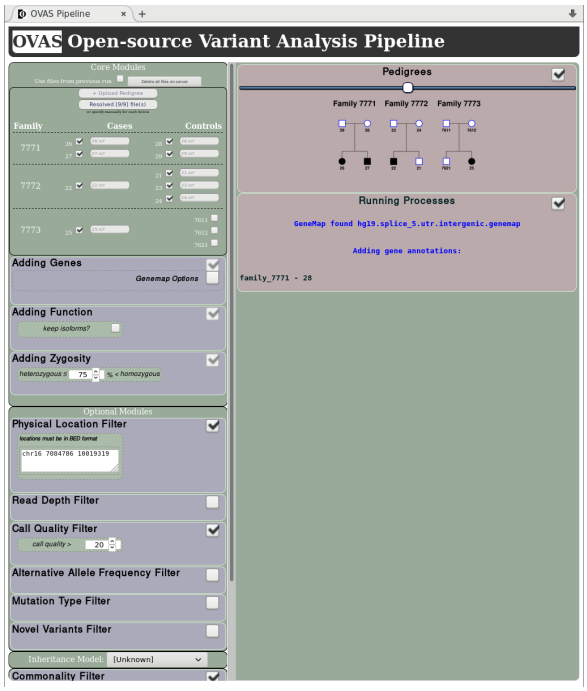

Figure 2 Web-interface displaying an ongoing analysis. The left sidebar shows the user-set configurations, and the central-right box displays the pedigrees used in the analysis stacked above a real-time progress box. Once complete, a summary will automatically open in a new browser tab.

Pipeline Stage	*Module Name*	*Runtime (seconds)*
Annotation	Adding Genes	125
	Adding Function	28.7
	Adding Zygosity	0.81
Filtering	Physical Location Filter	1.02
	Read Depth Filter	1.26
	Call Quality Filter	0.93
	AAF Filter	432
	Mutation Type Filter	1.08
	Novel Variant Filter	1.12
	Same Gene Filter	22.5
	Same Variant Filter	26.1
Trait Penetrance Model	AD Inheritance	0.83
	AR Inheritance	1.22
	XD Inheritance	0.74
	XR Inheritance	1.39
	Mosaicism	0.94
Extended Annotation	Isoform Context	2.28
	Protein Context	4.10
	Gene Expression	145

Table 1 Average single-core runtimes of VCF files containing 50,000 variants passing individually through all filters with timings for each Annotation, Filtering, and Extended Annotation modules. Trait Penetrance module timings are based on three VCFs consisting of a parent-offspring trio. Tests were run on a 2GHz dual-core processor with 4GB RAM.

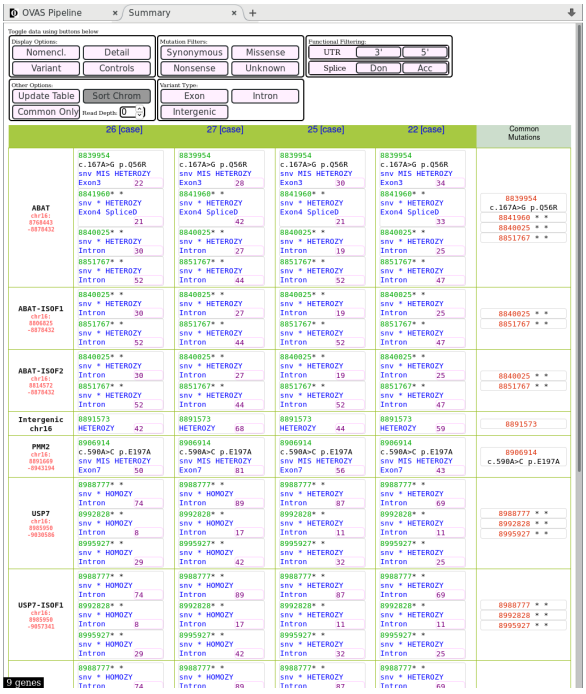

Figure 3 The summary tab which contains a comprehensive report of potential causative variants discovered in the analysis. The report is interactive and can perform dynamic filtering of discovered variants by toggling variants identified as synonymous, missense, nonsense, UTR (optionally 3' or 5'), and splice (optionally Donor or Acceptor). The promoter mutation for the case study is shown in the *Intergenic chr16* row.

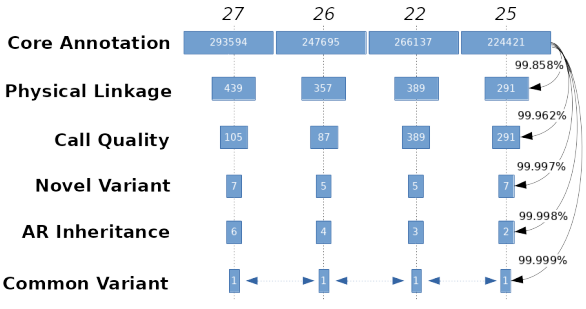

Figure 4 The progression of variants filtered at each subsequent annotation or filtering stage for each of the 4 case VCFs. Curved arrows display the average percentage reduction from the total number of variants.