Detection of aberrant events in RNA sequencing data

Vicente A. Yépez
Technical University of Munich https://orcid.org/0000-0001-7916-3643
Christian Mertes
Technical University of Munich
Michaela F. Mueller
Technical University of Munich
Daniela S. Andrade
Technical University of Munich
Leonhard Wachutka
Technical University of Munich
Laure Frésard
Stanford University https://orcid.org/0000-0001-8154-6328
Mirjana Gusic
Helmholtz Zentrum München
Ines Scheller
Technical University of Munich
Patricia F. Goldberg
Technical University of Munich
Holger Prokisch
Helmholtz Zentrum München https://orcid.org/0000-0003-2379-6286
Julien Gagneur (gagneur@in.tum.de)
Technical University of Munich https://orcid.org/0000-0002-8924-8365

Method Article

Keywords: RNA-seq, outlier detection, aberrant expression, aberrant splicing, mono-allelic expression, rare disorder, workflow

Posted Date: January 3rd, 2020

DOI: https://doi.org/10.21203/rs.2.19080/v1

Version of Record: A version of this preprint was published on January 18th, 2021. See the published version at https://doi.org/10.1038/s41596-020-00462-5.

Abstract

RNA sequencing (RNA-seq) has emerged as a powerful approach to discover disease-causing gene regulatory defects for individuals affected with a genetically undiagnosed rare disorder. Pioneer studies have shown that RNA-seq could increase diagnostic rates over DNA sequencing alone by 8% to 36 % depending on disease entities and probed tissues. To accelerate the adoption of RNA-seq among human genetic centers, detailed analysis protocols are now needed. Here, we present a step-by-step protocol that instructs how to robustly detect aberrant expression, aberrant splicing, and mono-allelic expression of a rare allele in RNA-seq data using dedicated statistical methods. We describe how to generate and assess quality control plots and interpret the analysis results. The protocol is based on DROP (Detection of RNA Outliers Pipeline), a modular computational workflow that integrates all analysis steps, can leverage parallel computing infrastructures, and generates browsable web page reports.

Introduction

Introduction

Whole-exome sequencing (WES) and whole-genome sequencing (WGS) are increasingly used to identify disease-causing genetic variants of individuals affected with a genetically undiagnosed rare disorder[1–3]. Depending on the disease entity, analysis of WES or WGS achieve a diagnosis rate of up to 50%[2–5]. One of the main challenges in these DNA-based molecular diagnostics approaches is the interpretation of the non-coding variants. Many of these variants may have a regulatory role in controlling RNA abundance or splicing. However, computational predictions of their regulatory effects remain uncertain[6]. To advance diagnostics, RNA sequencing (RNA-seq) has emerged as a complementary tool because it directly probes gene expression, thereby providing functional data supporting the clinical interpretation of variants. Moreover, RNA-seq can reveal genes with a regulatory defect, even in the absence of candidate regulatory variants, which is particularly important in the situation where WES instead of WGS has been performed. Pilot studies systematically using RNA-seq have obtained increased diagnostic rates over DNA-sequencing alone for a variety of rare disorders ranging between 8% and 36%[7–11].

As the first studies that systematically used RNA-seq for the genetic diagnosis of rare diseases were published only two years ago[7,8], the field is new and lacks established workflows. Here, we provide a step-by-step protocol that instructs how to use RNA-seq data in this context by supporting the detection of aberrant events. Specifically, we provide detailed instructions to identify aberrant expression, aberrant splicing, and mono-allelic expression (MAE) of a rare allele by starting from Binary Sequence Alignment Map (BAM)[12] and Variant Call Format (VCF) files[13] (Table 1). The protocol covers a number of quality control steps and plots that evaluate the counting and fitting procedures of each pattern. In order to control sample mixing, the protocol includes a validation step comparing variants called from RNA and DNA. Lastly, we describe how to analyze individual samples using the results and objects derived from the pipeline.

Application of the protocol

The aim of this protocol is to provide a one-in-all computational workflow starting from VCF and BAM files to call and visualize all forms of aberrant gene expression that have been currently used for rare disease diagnosis purposes. It is based on DROP (Detection of RNA Outliers Pipeline), a computational workflow that automatizes and standardizes all the steps needed to obtain the results from raw input files. The workflow uses the Snakemake framework[14] (Box 1) to guarantee reproducible results. Overall, our workflow can be easily implemented and provides a comprehensive collection of tools enabling the detection of outliers using RNA-seq.

To showcase the workflow, we have created a dataset based on 100 samples from the Geuvadis dataset[15] (Table S1), which is accessible without restriction under https://www.ebi.ac.uk/Tools/geuvadis-das/. We refer to this dataset as the demo dataset. DROP, including a small test dataset, is publicly available at https://github.com/gagneurlab/drop.

Advantages and limitations

Currently, only one computational workflow for RNA-seq based diagnostics of rare disorder has been published[9]. That pipeline includes alignment and quality controls of RNA-seq reads, functions to filter and annotate variants, and prioritize candidates. The main advantages of the workflow proposed here is that, using DROP, we automatize most of the steps, integrate the specialized methods OUTRIDER[16] and FRASER[17], add a module to test for MAE and VCF-BAM matching, include new quality control plots to evaluate the different steps, and render all results and plots in HTML pages. DROP is designed with a modular architecture, such that the aberrant expression, aberrant splicing, and MAE modules can be executed without depending on each other. Moreover, this architecture allows new modules with different functionalities to be added. Another advantage of DROP is that it uses the workflow management framework Snakemake[14] to run and monitor the execution of the pipeline (Box 1). Every time the pipeline is executed, Snakemake computes the dependencies among all scripts and data files. Then, it checks if either a data file or a script were modified or added. If a script is added or changed, Snakemake will execute it, together with the downstream steps. If a data file is added or changed, Snakemake will execute any script using it, and all the downstream scripts in cascade. Hence, this workflow management framework ensures that the results reflect the latest scripts and input data files while avoiding unnecessary executions of scripts lying upstream or parallel to the modifications.

Another advantage of DROP is the way in which it manages analysis groups. Users might not want to merge all the available samples into one analysis, but instead create subsets with, for example, samples extracted from the same tissue or belonging to the same disease entity. A sample can belong to one or

multiple groups described in the sample annotation. The groups are recognized by DROP and a separate analysis is created for each one of them.

RNA-seq is not the first tier diagnostic tool. Its power is in the detection of functional relevant consequences of DNA variants affecting gene expression levels and splicing. In this way it complements DNA sequencing by supporting interpretation of variants of uncertain significance and reprioritizing variants overlooked by DNA sequencing. RNA-seq has some inherent limitations in diagnostics (the highest reported diagnosis rate increases after WES using RNA-seq are 35%[8] and 36%[10]). One possible reason is that the gene is not expressed in the available tissue. Researchers can investigate a priori whether genes are expressed in a certain tissue by using public datasets like GTEx[18]. Another reason is that the expression or splicing are at normal levels in the available tissue, unlike in the affected tissue[10]. A final reason is that the disease-causing variant(s) (e.g. missense) might not have any effect on the transcript.

In order to confidently detect outliers, a large sample size is needed. The power analysis done in OUTRIDER concluded that the cohort should contain at least 50-60 samples[16]. This limitation can be overcome by integrating control samples from publicly available cohorts like GEUVADIS[15] or GTEx[18]. We advise to select case and control samples from the same tissues, and to align reads using the same aligner and parameters, before processing them jointly with DROP.

Assuming that local pipelines are established for alignment and variant calling, DROP takes as input BAM and VCF files (Table 1) and does not provide a built-in solution for those steps. However, this protocol provides in Materials instructions for those steps as well. Another limitation is that we describe only one method to compute the results of each type of aberrant expression pattern. Nevertheless, the user can use the count tables and modify the pipeline to apply any other method to them. Furthermore, this protocol does not include variant calling from RNA-seq, which has limited value when WGS is available, but is useful in the situation where only WES is available[8,10].

Workflow

Our workflow is composed of three main steps: i) preparing the input data, ii) fitting the models and extracting the results from aberrant expression, aberrant splicing, and MAE, and iii) analyzing the individual results (Fig. 1). The input data are raw BAM files, VCF files, a sample annotation table, a configuration file containing the pipeline's parameters, a human reference genome (FASTA) file, and a gene annotation (gtf) file (Table 1). DROP then generates intermediate files (e.g., read count matrices), final results, and produces HTML pages for convenient visualization using the framework wBuild (Box 1). The generated files are stored as either tab-separated text (tsv) files or binary R data (Rds) files, depending on their content. See Table 1 for file format overview. Finally, users can access the objects and results in order to plot and analyze the samples and genes of interest.

Aberrant expression

Aberrantly expressed genes, or expression outliers, refer to genes with an expression level aberrantly higher or lower in a sample with respect to other samples. Calling expression outliers in individuals affected with a rare disorder has been successfully used to identify disease-causing genes and to identify or confirm potential disease-causing variants[7,9,10].

To detect aberrant expression, we first compute an RNA-seq read count matrix by counting for each sample the sequencing reads that fully overlap each gene (Methods). Then, we apply OUTRIDER[16], a statistical method that computes significance level for extreme read count values while controlling for hidden confounders[16]. We extract *P* values, z scores and fold changes for each sample–gene combination from the OUTRIDER returned objects. After correcting for multiple testing, there is typically a handful of outlier genes per sample at a false discovery rate of 0.05. For example there is a median of 1 outlier per sample in the demo dataset, consistent with observations on other datasets[7,16]. If a sample has more than 0.1% of expressed genes as outliers, we refer to it as an aberrant sample. Pinpointing a candidate disease-causing genes among the tens to hundreds of expression outliers of an aberrant sample is difficult. Aberrant samples should be carefully analyzed to find a possible explanation for their high number of outliers, e.g., in case of contamination, or if the sample belongs to a different population (tissue, ethnicity). A biological cause is not to be excluded. For instance, disruption of the function of a transcription factor may cause aberrant expression of many downstream genes.

Other methods to detect aberrant expression consist of applying cutoffs on z scores[9,10] computed after regressing out contributions of hidden confounders estimated by PEER[19] or SVA[9]. However, simulations and enrichment analysis of rare variants among expresion outliers have shown that OUTRIDER outperformed these methods[16].

Aberrant splicing

Aberrant splicing is a major cause of Mendelian disorders[20,21]. Despite progresses in predicting aberrant splicing from sequence[22,23], the task remains difficult because splicing involves a complex set of cis-regulatory elements that are not yet fully understood. Moreover, some variants affecting splicing cannot be detected by WES for being located in intronic sequences not covered by WES kit probes. Aberrant splicing events can instead be advantageously detected from RNA-seq data[7–11,24–27].

The aberrant splicing module considers split reads that are reads aligning to separated location of a same chromosome and strand and therefore supportive of an exon-exon junction. This is done independently of exon annotation, allowing for calling splicing events de novo. In addition, reads

spanning the exon-intron boundary as well as the exon-intron boundary are counted to enable intron detection. The counts are converted into two intron-centric metrics, percent-spliced-in (PSI) and splicing efficiency (theta), which are calculated for both the donor *D* (5' splice site) and acceptor *A* (3' splice site) sites of every intron[28] (Methods). The theta_5 and theta_3 values are fitted together, hence we jointly refer to them as theta. Junctions expressed at low levels are filtered out (Methods). Then, for each of the metrics, the statistical method FRASER uses a denoising autoencoder to control for latent confounders and fits a beta-binomial distribution on every intron or splice site[17]. *P* values, z scores and *Delta Psi 5*, *Delta Psi 3*, and *Delta theta* values are then computed from the beta-binomial fits, where the *Delta* values are defined as the difference between the observed and the expected values. FRASER then reports gene-level *P* values as well as splice site-level *P* values. Similar to expression outliers, there is typically a handful of aberrant splicing events per sample that are significant at a FDR <0.05. For example, the median number of outlier events in the demo dataset are 2 for psi 5, 2 for psi 3, and 5 for theta.

Three other methods have been used to detect aberrant splicing. Specifically, (i) an adaptation of the differential splicing test LeafCutter[29] used in the Kremer et al. study[7], (ii) a cutoff based approach used in the studies of Cummings et al.[8] and Gonorazky et al.[10], and (iii) a z score based method used in the Frésard et al. study[9]. The first one constructs intron clusters and tests for differential usage between one sample and all others, instead of aberrant splicing events. Moreover, it does not control for sample covariation. The second one defines aberrant splicing events as novel introns in genes with enough reads in the affected individual but not (or almost not) appearing in a control cohort, after applying local normalization. The two main caveats are that it depends on arbitrary cut-offs and may fail to recognize aberrant splicing events in weak splice sites[30]. The third one, corrects for covariation, but uses a z score approach which does not offer any control for false discovery rate and can be inaccurate in splice sites with low reads. Having in mind these limitations, we have opted for FRASER[17], which addresses all those issues.

Mono-allelic expression

MAE refers to expression of a single allele out of the two alleles of a gene. When assuming a recessive mode of inheritance, single heterozygous rare variants are not prioritized after DNA sequencing. However, mono-allelic expression of such a single heterozygous rare variant in an affected individual, which could be due to genetic or epigenetic silencing of the other allele, is consistent with a recessive mode of inheritance. Therefore, detecting MAE of a rare variant has helped diagnosing rare disorders[7,9,10,26,31,32].

Detecting mono-allelically expressed genes relies on counting the reads aligned to each allele at genomic positions of heterozygous variants.

Several methods have been developed to detect MAE in the context of rare diseases, among which are the one described by Kremer et al.[7], and more recently ANEVA-DOT[31]. On the one hand Kremer et al. used a

negative binomial test with a fixed dispersion for all genes. On the other hand, ANEVA-DOT implements a binomial-logit-normal test with gene-specific variance *VG*, with the caveat that due to insufficient training data, estimates of *VG* have been computed so far for only 4,962 genes (in median), depending on the tissue of interest[31]. As using ANEVA-DOT would result in losing more than half of the tested genes, we opted for the training data-independent negative binomial test (Methods). In the demo dataset, we found 364 heterozygous SNVs to be mono-allelically expressed, 79 of which where toward the alternative allele, and 3 of which were rare (MAX AF $\leq$ 0.001).

VCF-BAM matching

A crucial step when performing multi-omics is to ascertain that all assays performed on samples obtained from the same individual correspond. Therefore, we designed an algorithm to match the variants derived from DNA and RNA sequencing based on the ideas proposed by t' Hoen et al.[33] and Lee et al.[34] (Methods). We achieved this by comparing the BAM files from RNA-Seq with the VCF files from DNA sequencing at predefined genomic positions of variants that are not in linkage disequilibrium. The proportion of variants derived from the same individual that match in the DNA and RNA has to be significantly higher than the one from different individuals, but will not reach 100% due to MAE[33]. The algorithm is applied not only to the annotated VCF-BAM samples, but to all combinations in order to find other possible unannotated matches.

In the demo dataset, the median of the proportion of matching genotypes of samples from the same individual is 0.98 compared to 0.58 in case of non matching samples. In the case of family members, we expect a value in-between these. All of the 100 RNA samples from the demo dataset match their corresponding DNA, as expected since it is a quality-controlled public dataset. This procedure is part of the MAE module, but can also be run independently. We provide a VCF file containing the set of genomic positions to be tested in our resource web server.

Detection of RNA Outliers Pipeline

To automatize and standardize the generation of the results, we developed the computational workflow DROP. DROP consists of three independent modules (Fig. 1). It is built using R and Python on top of the workflow manager frameworks Snakemake[14] and wBuild (Box 1). DROP offers a parallelized backend through Snakemake to execute the same processes on different samples at the same time, thus making effective use of computer clusters. It runs on the command line. Detailed instructions for installation, editing the input files and executing the pipeline can be found in the DROP documentation. The functions used throughout the workflow are explained in detail the Methods section.

Here we used a subset of 100 samples from the Geuvadis dataset to test our pipeline and generate the plots. We expect many groups to implement and benefit from our workflow, as using RNA-seq for rare

disease diagnosis is growing in popularity. Finally, as the source code is open source and the design of our pipeline is flexible, we encourage the community to expand the pipeline with new modules, e.g. by adding other multi-omics modules, prediction models, or calling variants on RNA-seq data.

Expertise needed

The installation, creation of a Python environment and set up of the config file should be done preferably by either a bioinformatician or a system administrator. Afterwards, the protocol can be followed by any scientist.

Box 1. Workflow Management Systems used in DROP

Snakemake (https://snakemake.readthedocs.io/en/stable) is a workflow management system to create data analyses guaranteeing reproducibility, automation, and scalability36. Snakemake workflows consist of rules that describe how to create output files from the respective input files. These output files are created by defining instructions in shell, Python or R code. Rule dependencies are determined bottom-up by matching file names with the support of automatically inferred named wildcards in the rules. Snakemake allows the usage of multiple scheduling systems or parallel backends (e.g., SGE, SLURM, multi core) to efficiently use HPC systems and run executions in parallel by automatically determining parallel parts in the workflow.

wBuild (https://wbuild.readthedocs.io) is a framework that automatically creates Snakemake dependencies, pipeline rules based on R markdown scripts and compiles the analysis results into a navigable HTML page. All information needed such as input, output, number of threads, and even Python code, is specified in a YAML header inside the R script file, thereby keeping code and dependencies together.

Figure Legends

Fig. 1 | Workflow overview. As input, DROP requires a configuration file, a sample annotation file, BAM files from RNA-seq, and VCF files. DROP processes then for each module the input data and generates count tables, overview plots (e.g. sample covariation heatmap), quality control plots, and result tables. Lastly, users can perform case-by-case analysis with the help of different visualizations.

Fig. 2 | The Aberrant Expression Module. a, Histogram of raw read count distribution colored by different filtering strategies: all, minimum 1 count in 5% of the samples, minimum 10 counts in 5% of the samples, and minimum 1 (or user-defined) FPKM in 5% of the samples. **b,** Number of expressed genes cumulative across all samples. Colors represent the union of all detected genes (blue), genes that passed the FPKM

filter as a group (violet), genes that are expressed on each sample (red), and the intersection of expressed genes (green). **c,** Heatmap of the correlation of row-centered log-transformed read counts between samples before and after (**d**) the autoencoder correction including the samples' sexes and the laboratories where they were sequenced. Before correction, samples cluster by laboratory.

Fig. 3 | The aberrant splicing module. a, Histogram showing the junctions that passed the filter (Methods). **b,** Number of aberrant splicing events per sample by gene colored by metric. **c,** Heatmap of the correlation of row-centered logit-transformed read count ratios between samples before and after (**d**) the correction.

Fig. 4 | Mono-allelic Expression Module. a, Cascade plot showing the number of heterozygous SNVs with at least 10 reads (median=11,532), that are mono-allelically expressed (median=364), mono-allelically expressed toward the alternative allele (median=79) and rare (median=3). The lower and upper limits of the boxes correspond to the first and third quartiles, respectively. Whiskers extend to the most extreme value, no further than 1.5x the interquartile range. **b,** Histogram (y-axis in log scale) of the proportion of matching genotypes from DNA and RNA when comparing all sample combinations. The median of matching samples is 0.98 compared to 0.58 of non-matching samples. One sample had a very low rate of matching with the rest of samples (median=0.26).

Fig. 5 | Analysis plots. a, Negative log-transformed nominal *P* values (y-axis) versus z scores (x-axis) derived from the expression of all genes of sample NA20804 showing one over- and one underexpression outlier (red). **b,** Normalized counts (y-axis) of gene *MUTYH* of all samples (x-axis) reflecting one underexpression outlier (red). **c,** Representative sashimi plot showing the creation of a new exon for Sample 1 which is skipped by the other two samples. The height represents the coverage in log10 RPKM and the number of the arcs the junction's coverage. On the bottom, the corresponding gene model is depicted (in red the cryptic exon). **d,** Fold change between the counts of the alternative and reference alleles (y-axis) compared against the total coverage of each heterozygous SNV (with at least 10 reads) of sample HG00096, highlighted by significance (orange), and significance and rarity (red).

Table Legends

Table 1 | File formats

Table 2 | Files created by DROP and their locations. The <root>, <htmlOutputPath>, and <geneAnnotation> variables are defined in the config file. The <group> variable corresponds to the DROP_GROUP column defined in the sample annotation.

Reagents

BAM files from RNA-seq data

This workflow focuses on the analysis of the sequencing data rather than on their generation and preprocessing. In this subsection we explain how the BAM files (Table 1) should be generated to serve as input for DROP. BAM files are created by aligning FASTA files derived from RNA-seq to a reference genome. We recommend using the aligner STAR36 with the default parameters and twopassMode = 'Basic' to detect novel splice junctions. We strongly recommend performing quality control with tools like MultiQC35. The BAM files contain reads that will be used in all of the modules to generate the read count matrices. They do not need to be in the same folder, as the location is specified in the sample annotation table. The BAM files must be sorted by position and indexed. This can be achieved using the commands sort -o and index from SAMtools34.

VCF files from either WES or WGS

VCF files (Table 1) are generated through calling variants on a BAM file. Therefore, the FASTA files derived from DNA sequencing must be aligned to a reference genome first (one possible tool is the Burrows-Wheeler Aligner39). BAM files must be sorted by position and indexed. Afterwards, variants can be called by following, for example, the GATK guidelines37,38. This will generate either one VCF file per DNA sample, or a single VCF file containing multiple samples. DROP can handle either option. VCF files must be compressed and indexed. This can be achieved by using the bgzip -c and tabix -p commands from Tabix39, respectively. They do not need to be stored in the same folder, as the location is specified in the sample annotation table. We further recommend annotating the VCF file(s) with the Variant Effect Predictor32 following their tutorial (https://uswest.ensembl.org/info/docs/tools/vep/script/vep_tutorial.html). This step will add information such as variant consequences (e.g. missense, stop-gained) and allele frequencies.

Human reference genome file

In order to compute the allelic counts, a human reference genome (FASTA) file is needed. Preferably, the file should be the same that was used to align the samples' FASTA files. Otherwise, the user can download it from different sources, as long as the genome build (either GRCh37 (hg19), or GRCh38 (hg38)) matches the BAM files. For example, for build 19 https://hgdownload.soe.ucsc.edu/goldenPath/hg19/bigZips/, hg19.fa.gz; and for 38 https://hgdownload.soe.ucsc.edu/goldenPath/hg38/bigZips/latest/, hg38.fa.gz, from UCSC40. Finally, an index (.fai) file must be created in the same directory where the FASTA file is located using the function faidx from SAMtools12.

Gene annotation file

In order to count the reads from the BAM files in the aberrant expression module, a gene annotation file is required. DROP was developed using the 'main annotation' gtf file from the release 29 of the GENCODE annotation41. Nevertheless, newer releases already exist and will continue to appear. As gtf files from other sources can have other columns and with different names, DROP might not be able to parse them. Therefore, to avoid problems, we recommend users to download the latest release from GENCODE, with the right genome build (https://www.gencodegenes.org/human/).

Equipment

Hardware

A multi-core computer with at least 16 cores and 64 GB RAM is suggested, but depends heavily on the analyzed dataset (e.g., sample size). For developing and testing DROP on the 100 samples from the Geuvadis dataset, a Scientific Linux 7.7 based server with 32 physical cores and 256 GB memory was used.

Software

- A Linux based operating system

- DROP (https://github.com/gagneurlab/drop)

- Python, version 3.6.7 or higher

- We recommend using a virtual environment such as conda (https://docs.conda.io/) for making sure that the correct versions are installed

- Pip version 19.1 or higher

- R (https://www.r-project.org/), version 3.5 or higher

Bioconductor (https://bioconda.github.io/) and devtools need to be installed. Please make sure that the library paths are writable.

- The required packages are listed in the DROP repository and an installation script is provided.

- An integrated development environment (IDE) like RStudio (https://www.rstudio.com/products/rstudio/download/) is recommended.

- GATK37 (https://software.broadinstitute.org/gatk/).

- SAMtools12 version 1.7 or higher and Tabix39 can be both installed from HTSlib. (https://www.htslib.org/download/).

- BCFtools42. Use the latest version from its github page (https://github.com/samtools/bcftools).

- Integrative Genomics Viewer (IGV)43 (https://software.broadinstitute.org/software/igv/)

- Graphviz (https://www.graphviz.org/)

- Pandoc (https://pandoc.org/)

Procedure

Configuration of the workflow

1. Define a directory where the analysis code will be. Go to that directory and open a terminal. Set up a new DROP-based project by executing *drop init*. This will create a config file, a Snakefile and a Scripts directory for analysis that can be modified by the user.

2. Adjust the keys of the config.yaml file, which follows the YAML format44. All non-file keys contain default values that can but do not have to be changed. A detailed description of parameters and file names can be found in the DROP documentation.

3. Create the sample annotation table. Each row corresponds to a unique pair of RNA and DNA samples derived from the same individual. An RNA assay can belong to one or more DNA assays, and vice-versa. If so, they must be specified in different rows. The required columns are RNA_ID, RNA_BAM_FILE and DROP_GROUP, plus other module-specific ones. Save it in the tab-separated values format (Table 1). The column order does not matter. Also, it does not matter where it is stored, as the path is specified in the config file. An example is shown in Table S1. Further instructions and examples can be found in the DROP documentation. ?TROUBLESHOOT

RNA_ID: unique identifier from an RNA assay.

RNA_BAM_FILE: absolute path of the BAM file derived from RNA-seq (e.g. /home/project1/Data/RNA1.bam, and not RNA1.bam). A BAM file can only belong to one RNA_ID, and vice-versa.

DROP_GROUP: the analysis group(s) that the RNA assay belongs to. Multiple groups must be separated by commas (e.g., blood,WES,all). We recommend doing a different analysis for each tissue as gene expression and splicing can be tissue-specific.

MAE module specific columns:

DNA_ID: unique identifier from a DNA assay.

DNA_VCF_FILE: absolute path to the corresponding VCF file (e.g., /home/project1/Data/sample1.vcf.gz, and not sample1.vcf). The DNA_ID has to match the ID inside the VCF file. In case a multi-sample VCF is used, write the file name for each sample.

The following columns describe the RNA-seq experimental set-up. They affect the counting procedures of the aberrant expression and splicing modules. For a detailed explanation, refer to the documentation of HTSeq45.

PAIRED_END: either TRUE or FALSE, depending on whether the sample comes from paired-end RNA-seq or not.

STRAND: either yes, no or reverse: “no” means that the sequencing was not strand-specific; “yes” that it was strand-specific, and the first read in the pair is on the same strand as the feature and the second read on the opposite strand; and “reverse” that these rules are reversed.

The following columns describe how the counting should be performed in the aberrant expression module. We recommend setting the count mode to IntersectionStrict to consider only reads that fully align within the gene, and the count overlaps to TRUE to detect as many genes as possible. Refer to the documentation of HTSeq45 for details.

COUNT_MODE: either “Union”, “IntersectionStrict” or “IntersectionNotEmpty”.

COUNT_OVERLAPS: either TRUE or FALSE, depending on whether reads overlapping different regions are allowed and counted.

4. Execute *snakemake sampleAnnotation*, which will parse and check the provided sample annotation and open up the annotation overview in a report webpage (Table 2). The annotation overview contains different information regarding the sample annotation and different sanity checks. Go through it carefully and, if found, correct any errors. Check if the number of samples of each DROP group are correct.

5. Execute *snakemake -n*. It will display all the jobs that must be performed. The full pipeline can already by run by executing *snakemake <options>*. Nevertheless, we recommend to run it by parts, especially the first time. For a detailed explanation of the “options”, check the DROP documentation.

The aberrant expression module

6. To run the aberrant expression module execute *snakemake <options> aberrantExpression*. This step is computationally heavy and can take several hours. This command counts the reads for each sample (Methods), performs the OUTRIDER fit, extracts the results, and creates two HTML reports with different plots for counting and results (Fig. S1a). This is performed for each annotation and group combination defined in the geneAnnotation parameter (config file) and in the DROP_GROUP column (annotation file),

respectively. Different intermediate and final objects are saved locally (Table 2) and can be used for further analyses.

7. To look at the counting outcome, open the aberrant expression overview webpage (Table 2). Go to the Counting tab. It contains one link per analysis group. Click on the analysis group you want to visualize. It contains the following quality control plots:

- Percentage of reads counted per sample (Fig. S1b).

- Number of reads counted per sample.

- Size factors per sample (Fig. S1c).

- Raw read count distribution (Fig. 2a).

- Number of expressed genes (Fig. 2b). A total of 17,259 genes passed the filter in the demo dataset.

8. To investigate the results, open the overview webpage. Go to the OUTRIDER tab. It contains one link per analysis group. Click on the analysis group you want to visualize. It contains the following:

- Fit of the encoding dimension (Fig. S1d).

- Number of aberrant genes per sample (Fig. S1e).

- Heatmaps showing the sample correlation before (Fig. 2c) and after the correction (Fig. 2d). The clustering should disappear after the correction.

- Heatmaps showing the log row-centered gene expression of the 50 most variable genes before and after the correction. This is useful to identify the genes that drive the clustering of samples.

- Gene-wise biological coefficient of variation46 before and after the correction.

- Table with aberrant samples.

- OUTRIDER results table with the option to be filtered (subsetted for significant events only) and links to download both the full and subsetted results tables.

The aberrant splicing module

9. To run the aberrant splicing module execute *snakemake <options> aberrantSplicing*. This step is also computationally heavy and can take several hours. It creates a FraseR dataset object that contains all the intron-centric metrics and results, an HTML page with overview plots, and the splicing outlier results table (Fig. S2a).

10. Open the overview webpage. Click on the analysis group you want to visualize. It contains the following:

- Filter expression plot (Fig. 3a)

- Number of aberrant events per sample (Fig. 3b).

- Heatmaps (sample vs. sample) before (Fig. 3c) and after correction (Fig. 3d).

- Finding best encoding dimension for psi3 (Fig. S2b), psi5(Fig. S2c), and theta.

- Results searchable table for all metrics combined.

Mono-allelic Expression Module

11. To run the MAE module execute *snakemake <options> MAE*. This creates the allelic counts, aggregated results table (Methods) and the overview webpage (Fig. S3). The results for each sample contain the counts, the nominal and the adjusted *P* values, and alternative allele ratio (altRatio = #ALT / (#ALT+#REF)) of each variant. Moreover, the results are annotated with the minor allele frequencies from gnomAD47 (for each of the African, American, East Asian and Non-Finnish European populations, and the maximal frequency among them), and a boolean column “rare” if the allele frequency was below the corresponding cut-off specified in the config file. This step is computationally heavy and can take several hours.

12. Open the MAE overview webpage (Table 2). Go to the MAE tab. It contains different summarizing information:

- cascade plot (Fig. 4a) showing the number of heterozygous SNVs for the four cumulative filter criteria: at least 10 reads, mono-allelically expressed for either the reference or alternative allele, for the alternative only, and rare.

- searchable results table containing only the significant events. To subset for rare variants, simply filter the “rare” column.

- links to download full and significant-only results tables.

VCF - BAM matching

13. Download the test VCF file (qc_vcf_1000G.vcf.gz) and its index file from the DROP resource webpage (Online resources). Its location is specified in the config file.

14. To run the VCF - BAM file matching, execute *snakemake <options> sampleQC*. This step is computationally heavy and can take several hours.

15. Open the MAE overview webpage (Table 2). Go to the QC tab. It contains:

- Histogram of the percentage of matching genotypes between DNA and RNA (Fig. 4b).

- Table containing the samples that were annotated to match, but actually do not (empty if there are no mismatches).

- Table containing the samples that were not annotated to match, but actually do (empty if there are no such cases).

Exploration of outlier results

16. Execute *snakemake <options>*. This will run all the analysis scripts and create a webpage for each of them with case-specific plots.

17. Open the aberrant expression analysis webpage (Table 2). It contains the following:

- Links to the overview webpages, and the paths of the OUTRIDER dataset objects and results tables.

- Volcano plot (-log10(*P* value) vs. z score) of the first sample from the results table. Figure. 5a shows a typical scenario in which the expression of most of the genes of a sample does not significantly deviate from the rest of the population with the exception of a handful of genes clearly distinguishable from the other ones.

- Expression plot (normalized counts across all samples) of the first gene from the results table (Fig. 5b). This plot not only shows if the gene of interest is aberrantly expressed in a sample, but also the magnitude. In this case, the expression of gene *MUTYH* lies in a range of 350-500 counts for all samples except for one, where it is around 200 counts, thus implying a ~50% reduction.

- Expected vs. observed counts plot of the same gene across all samples (Fig. S4a). This plot emphasizes the need to correct the raw reads, as samples with low reads can anyway have a low expected value after the OUTRIDER fit.

18. Open the aberrant expression analysis script (Table 2) in your prefered editor (e.g., RStudio). It contains the code that generated the analysis webpage. Modify the code by specifying a sample and gene of interest and recreate the plots from the previous step by executing the different lines of code. Alternatively, save the script and execute *snakemake <options>* to generate an updated analysis webpage.

19. Open the aberrant splicing analysis webpage (Table 2). It contains the following:

- Links to the overview webpages, and the paths of the FRASER dataset objects and results tables.

- Volcano plot (-log10(*P* value) vs.) of the first sample from the results table (Fig. S4b).

- Expression plot (junction count versus total junction coverage) for 3 of the first junction from the results table across all samples (Fig. S4c). In this example, the count of the junction was the same as the total count of the corresponding acceptor, in most of the samples. Nevertheless, in one sample, the total count for the acceptor is around 3 times higher than the tested junction. This means that most of its spliced reads come from another donor not present in any other sample.

- Observed vs. predicted counts plot for 3 of the same junction across all samples (Fig. S4d).

- Quantile-quantile plot of observed vs. expected *P* values for 3 of the same junction (Fig. S4e) to evaluate the fit.

20. Open the aberrant splicing analysis script (Table 2) in RStudio. Once again, go through the different lines of code and modify them to specify the sample and junction of interest. Recreate the plots. Run *snakemake <options>* to display the plots in the analysis webpage. These plots indicate if an event was an outlier by visualizing all samples. Nevertheless, sashimi plots provide a better visualization of the nature of the splicing aberration.

21. To create a sashimi plot, load the BAM file of the sample of interest and also other BAM files that do not have this splicing outlier in IGV43 (File > Load from File, then select the BAM file). Go to the genomic position of the junction of interest obtained from the FRASER results table. Adjust the zoom to cover the entire region of interest and create a sashimi plot by doing right click > Sashimi Plot. The bars represent the coverage for each alignment track, and the arcs the splice junctions connecting exons with the number of reads split across the junction. Therefore, sashimi plots can be used for both qualitative and quantitative analysis of transcripts. High-quality sashimi plots (Fig. 5c) can be achieved using the MISO package51. We provide an example script, config and gff file needed to generate a MISO sashimi plot in the DROP github webpage.

22. Open the MAE analysis webpage (Table 2). It contains the following:

- Links to the overview webpages, and the paths of the results per sample.

- MA plot (fold change vs. RNA coverage) of the first sample from the results table, colored by significance and rarity (Fig. 5d). As expected, a lower proportion of variants are mono-allelically expressed towards the alternative than towards the reference, and from them only a handful are rare.

- Alternative vs. reference allele coverage plot of the same sample (Fig. S4f), which gives a similar overview as the previous plot.

23. Open the MAE analysis script (Table 2) in RStudio. Once again, go through the different lines of code and modify them to specify the sample of interest. Note that the format of the identifier is {RNA_ID}–

{VCF_ID}. Recreate the plots. Run snakemake <options> to display the plots in the analysis webpage.

24. Mono-allelic expression can also be inspected in IGV. Load the BAM file of interest in IGV. If available, load the corresponding WES or WGS BAM file, and other BAM files as controls. Go to the position of the mono-allelically expressed variant (from the MAE results table) and confirm the detected mono-allelic expression.

Adding or editing input data

25. Every time a new input file is added or edited the user should execute *snakemake -n*. DROP will automatically identify the steps that need to be run to obtain the new results. Run each of the modules and analysis scripts as done before. Be sure to include any new sample in the sample annotation table. Alternatively, executing *snakemake <options>* runs the full pipeline.

26. Changes in the config file and in the sample annotation (except for added rows) are not recognized by DROP. Therefore, executing *snakemake -n* will reflect no new jobs to be performed. A way to force one of the modules to be executed is by running *snakemake –forcerun <module>*.

Troubleshooting

Step 5:

Problem: The sample annotation does not load properly.

Possible reason: There were problems with parsing special values

Solutions 1. Avoid special characters. 2. The DROP_group column has comma separated values, so be sure to save the sample annotation with tab-separated values (tsv). 3. Go through the sample_annotation html report and try to find any inconsistency.

Step 7:

Problem: The gene counts are very low

Possible reason: Be sure about the RNA-seq protocol of your samples to determine whether they were strand-specific and to which strand the first read aligns to

Solutions: Load your samples in IGV and color them by first-of-pair strand. Go to a gene that lies on the forward strand. Scroll over the reads and note the “Pair orientation” value. Set the STRAND column from the sample annotation to:

- “no” if the reads have multiple colors

- “yes” if the “Pair orientation” is F1R2

- “reverse” if the “Pair orientation” is F2R1

Step 7:

Problem: Very few genes are counted

Possible reason:

- A gtf file from a different genome build was used.

- Too low coverage.

Solutions:

- Be sure to provide a gtf file with the same build (hg 19 or hg38) as used for aligning the BAM files.

- Check the coverage of the BAM files.

Time Taken

The timings below are provided for the demo dataset (composed of 100 samples) with 20 available CPU cores and 64GB RAM. However, the performance highly depends on the dataset size and the number of available CPU cores and RAM.

Installation: 3 h max if all dependencies must be installed

Creating the sample annotation: ~2 h depending on experimental design and annotations at hand

Modifying the keys of the config file: 15 min.

Aberrant expression module: 20 min counting/sample/core, 6 h OUTRIDER fit and results.

Aberrant splicing module: 4 h for counting and 3 h for fitting and generating the results.

MAE module: 2 h counting/sample, 1 h for generation of all results.

Sample QC module: same as MAE module.

Anticipated Results

Results

One can expect a handful of expression outliers per sample. In the demo dataset the median number of expression outliers is 1 and 90% of samples had 7 expression outliers (Fig. S1e), in line with previous reports[7,16] and other in-house analyses. These expression outliers should be interpreted according to their fold changes. A very strong down-regulation (fold-changes < 0.2) very probably implies impaired gene function. Fold-changes of weaker amplitudes should further be investigated. In particular a fold-change of 0.5 often reflects loss of expression of one allele. Inspecting the MAE results may reveal mono-allelic expression of a rare variant harbored by the other allele[7,10].

One can also expect a handful of splicing outliers per sample. These aberrant splicing events should be visualized as sashimi plots, for instance using the genome browser IGV to detect the nature of the splicing defect including intron retention, exon truncation, exon elongation, or exon skipping. The next step is to look for direct splice site and near splice site variants that can explain the defect. However, other variants such as synonymous and deep intronic, have also been described to activate new splice sites[9] potentially originating from cryptic exons[7,8,10,11].

Analysis of RNA-seq deliver candidate genes whose aberrant expression or splicing may be contributing to the clinical presentation of the patient. To prioritize candidate genes, useful complementary information includes disease gene list such as OMIM[48] and whether gene function matches the patient phenotypes as done, for example, in Frésard et al.[9]. Promising candidate genes need to be further investigated for potential pathogenic variants. RNA-seq informs about consequences of genetic variants but not necessarily allows pinpointing the causal variant. Nonetheless, in some instances a splice defect is directly explained by a variant in the splice site. Also, underexpression outliers can be explained by nonsense mediated decay triggered by premature termination codons caused by stop-gain variants or frameshift variants. In this respect the value of RNA-seq is to support prioritization of candidate genes which need further inspection or functional validation of an already candidate variant. In any case, for the molecular diagnostic interpretation we refer to the ACMG standards and guidelines for the interpretation of sequence variants[49].

Methods

Aberrant expression

To generate the gene counts, DROP uses the summarizeOverlaps function from the GenomicAlignments package[50]. The parameters mode, singleEnd, ignore.strand, and inter.feature are all specified by the user in the config file (refer to main text or HT-Seq[45] documentation). Afterwards, genes where at least 5% of the samples have a FPKM > fpkmCutoff (defined in the config file) are filtered out. Finally, the OUTRIDER fit is run for each analysis group and the results tables are created.

Aberrant splicing

The whole module is implemented as described in FRASER17. Briefly, split reads are counted using the summarizeJunctions function from the GenomicAlignments package50, and non-split reads overlapping splice sites are counted using the featureCounts function from the Rsubread package51. Then, they are converted into the intron-centric metrics percent-spliced-in and splicing efficiency28. The percent-spliced-in index is computed as the ratio between reads mapping to the given intron and all split-reads sharing the same donor or acceptor site. To detect partial or full intron retention, the splicing efficiency metric is used. It is defined as the ratio of all split-reads and the full read coverage at a given splice site.

Afterwards, introns with less than 20 reads in all samples and introns for which the total number of reads at the donor and acceptor splice site is zero in more than 5% of the samples are filtered out. Lastly, the FRASER fit is run and the results are extracted.

Mono-allelic expression

First, DROP subsets the VCF(s) files to obtain only SNVs using the view command from bcftools42. Then, the allelic counting is performed using the ASEReadCounter52 function from GATK37. The negative binomial test is applied on the reads using the DESeq2 package53 fixing the dispersion parameter to 0.05 as done in Kremer et al.7. Finally, allele frequencies from gnomAD47 are added using the packages MafDb.gnomAD.r2.1.hs37d5 and MafDb.gnomAD.r2.1.GRCh38.

VCF-BAM matching

In order to match the variants from DNA and RNA, first a set of variants that are not in linkage disequilibrium (as correlated variants can bias the results54) is needed. In order to obtain that set, we pooled all of the variants from the samples in the demo dataset and consider only the ones in autosomal chromosomes that are not in linkage disequilibrium using the function snpgdsLDpruning from the R/Bioconductor package SNPRelate55. Applying a linkage disequilibrium threshold of 0.2, we obtained a set of P = 26,402 variants and their genomic positions.

For each of the N VCF files, we check for variants at those positions, thus generating a vector xi =[0/0, 0/1, 1/1, …,] of size P, where 0/0 represents no variant, 0/1 heterozygous, 1/1 homozygous variant, and $i=1,…,N$ is a counter for the VCF files. Then, we compute the allelic counts at those P positions using all M BAM files. We test if they are mono-allelically expressed and obtain a vector yj =[NA, 0/1, 1/1, 0/0, …,] of size P, where 0/0 means a ratio of the alternative allele (ratioALT) < 0.2, 1/1 that ratioALT > 0.8, 0/1 that 0.2 ≤ ratioALT ≤ 0.8, and NA that the position was not expressed or had less than 10 reads, and $j=1,…,M$

is a counter for the BAM files. Finally, we count the number of elements that are the same for each combination of vectors *xi* , *yj* and divide it by the length of *yj* after removing missing values, thus generating an $N \times M$ matrix.

Online resources

DROP documentation: https://drop-rna.readthedocs.io/en/latest/

DROP in github: https://github.com/gagneurlab/DROP

DROP resource folder: https://i12g-gagneurweb.in.tum.de/public/paper/drop_analysis/resource/

Candidate prioritization: https://github.com/lfresard/toolbox/blob/master/Candidate_Gene_Prioritization.ipynb

FraseR in github: https://github.com/gagneurlab/FraseR

Geuvadis dataset: https://www.ebi.ac.uk/Tools/geuvadis-das/

HTSeq documentation: https://htseq.readthedocs.io/en/release_0.11.1/count.html

OUTRIDER in Bioconductor: https://bioconductor.org/packages/release/bioc/html/OUTRIDER.html

Snakemake documentation: https://snakemake.readthedocs.io/en/stable/

wBuild documentation: https://wbuild.readthedocs.io/en/latest/

YAML documentation: https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html

References

1. Bamshad, M. J. *et al.* Exome sequencing as a tool for Mendelian disease gene discovery. *Nat. Rev. Genet.* **12**, 745–755 (2011).

2. Yang, Y. *et al.* Clinical Whole-Exome Sequencing for the Diagnosis of Mendelian Disorders. *N. Engl. J. Med.* **369**, 1502–1511 (2013).

3. Taylor, J. C. *et al.* Factors influencing success of clinical genome sequencing across a broad spectrum of disorders. *Nat. Genet.* **47**, 717–726 (2015).

4. Lionel, A. C. *et al.* Improved diagnostic yield compared with targeted gene sequencing panels suggests a role for whole-genome sequencing as a first-tier genetic test. *Genet. Med.* **20**, 435–443 (2018).

5. Chong, J. X. *et al.* The Genetic Basis of Mendelian Phenotypes: Discoveries, Challenges, and Opportunities. *Am. J. Hum. Genet.* **97**, 199–215 (2015).

6. Cooper, G. M. Parlez-vous VUS? *Cold Spring Harb. Lab. Press* 1423–1426 (2015) doi:doi/10.1101/gr.190116.115.

7. Kremer, L. S. *et al.* Genetic diagnosis of Mendelian disorders via RNA sequencing. *Nat. Commun.* **8**, (2017).

8. Cummings, B. B. *et al.* Improving genetic diagnosis in Mendelian disease with transcriptome sequencing. *Sci. Transl. Med.* **9**, 12 (2017).

9. Frésard, L. *et al.* Identification of rare-disease genes using blood transcriptome sequencing and large control cohorts. *Nat. Med.* **25**, 911–919 (2019).

10. Gonorazky, H. D. *et al.* Expanding the Boundaries of RNA Sequencing as a Diagnostic Tool for Rare Mendelian Disease. *Am. J. Hum. Genet.* **104**, 466–483 (2019).

11. Lee, H. *et al.* Diagnostic utility of transcriptome sequencing for rare Mendelian diseases. *Genet. Med.* (2019) doi:10.1038/s41436-019-0672-1.

12. Li, H. *et al.* The Sequence Alignment/Map format and SAMtools. *Bioinformatics* **25**, 2078–2079 (2009).

13. Danecek, P. *et al.* The variant call format and VCFtools. *Bioinformatics* **27**, 2156–2158 (2011).

14. Köster, J. & Rahmann, S. Snakemake--a scalable bioinformatics workflow engine. *Bioinformatics* **28**, 2520–2522 (2012).

15. Lappalainen, T. *et al.* Transcriptome and genome sequencing uncovers functional variation in humans. *Nature* **501**, 506–511 (2013).

16. Brechtmann, F. *et al.* OUTRIDER: A Statistical Method for Detecting Aberrantly Expressed Genes in RNA Sequencing Data. *Am. J. Hum. Genet.* **103**, 907–917 (2018).

17. Mertes, C. *et al. Detection of aberrant splicing events in RNA-Seq data with FRASER*. https://tinyurl.com/FRASER-paper (2019).

18. GTEx Consortium. Genetic effects on gene expression across human tissues. *Nature* **550**, 204–213 (2017).

19. Stegle, O., Parts, L., Piipari, M., Winn, J. & Durbin, R. Using probabilistic estimation of expression residuals (PEER) to obtain increased power and interpretability of gene expression analyses. *Nat. Protoc.* **7**, 500–507 (2012).

20. Scotti, M. M. & Swanson, M. S. RNA mis-splicing in disease. *Nat. Rev. Genet.* **17**, 19–32 (2016).

21. Singh, R. K. & Cooper, T. A. Pre-mRNA splicing in disease and therapeutics. *Trends Mol. Med.* **18**, 472–482 (2012).

22. Cheng, J. *et al.* MMSplice: modular modeling improves the predictions of genetic variant effects on splicing. *Genome Biol.* **20**, 48 (2019).

23. Jaganathan, K. *et al.* Predicting Splicing from Primary Sequence with Deep Learning. *Cell* **176**, 535-548.e24 (2019).

24. Gonorazky, H. *et al.* RNAseq analysis for the diagnosis of muscular dystrophy. *Ann. Clin. Transl. Neurol.* **3**, 55–60 (2016).

25. Kernohan, K. D. *et al.* Whole-transcriptome sequencing in blood provides a diagnosis of spinal muscular atrophy with progressive myoclonic epilepsy. *Hum. Mutat.* **38**, 611–614 (2017).

26. Hamanaka, K. *et al.* RNA sequencing solved the most common but unrecognized NEB pathogenic variant in Japanese nemaline myopathy. *Genet. Med.* **21**, 1629–1638 (2019).

27. Wang, K. *et al.* Whole-genome DNA/RNA sequencing identifies truncating mutations in RBCK1 in a novel Mendelian disease with neuromuscular and cardiac involvement. *Genome Med.* **5**, (2013).

28. Pervouchine, D. D., Knowles, D. G. & Guigo, R. Intron-centric estimation of alternative splicing from RNA-seq data. *Bioinformatics* **29**, 273–274 (2013).

29. Li, Y. I. *et al.* Annotation-free quantification of RNA splicing using LeafCutter. *Nat. Genet.* **50**, 151–158 (2018).

30. Kapustin, Y. *et al.* Cryptic splice sites and split genes. *Nucleic Acids Res.* **39**, 5837–5844 (2011).

31. Mohammadi, P. *et al.* Genetic regulatory variation in populations informs transcriptome analysis in rare disease. *Science* **366**, 351–356 (2019).

32. Albers, C. A. *et al.* Compound inheritance of a low-frequency regulatory SNP and a rare null mutation in exon-junction complex subunit RBM8A causes TAR syndrome. *Nat. Genet.* **44**, 435–439 (2012).

33. 't Hoen, P. A. C. *et al.* Reproducibility of high-throughput mRNA and small RNA sequencing across laboratories. *Nat. Biotechnol.* **31**, 1015–1022 (2013).

34. Lee, S. *et al.* NGSCheckMate: software for validating sample identity in next-generation sequencing studies within and across data types. *Nucleic Acids Res.* **45**, (2017).

35. Ewels, P., Magnusson, M., Lundin, S. & Käller, M. MultiQC: summarize analysis results for multiple tools and samples in a single report. *Bioinformatics* **32**, 3047–3048 (2016).

36. Li, H. & Durbin, R. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics* **25**, 1754–1760 (2009).

37. McKenna, A. *et al.* The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Res.* **20**, 1297–1303 (2010).

38. Van der Auwera, G. A. *et al.* From FastQ Data to High-Confidence Variant Calls: The Genome Analysis Toolkit Best Practices Pipeline. in *Current Protocols in Bioinformatics* 11.10.1-11.10.33 (John Wiley & Sons, Inc., 2013). doi:10.1002/0471250953.bi1110s43.

39. Li, H. Tabix: fast retrieval of sequence features from generic TAB-delimited files. *Bioinformatics* **27**, 718–719 (2011).

40. Haeussler, M. *et al.* The UCSC Genome Browser database: 2019 update. *Nucleic Acids Res.* **47**, D853–D858 (2019).

41. Frankish, A. *et al.* GENCODE reference annotation for the human and mouse genomes. *Nucleic Acids Res.* **47**, D766–D773 (2019).

42. Li, H. A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. *Bioinformatics* **27**, 2987–2993 (2011).

43. Robinson, J. T. *et al.* Integrative genomics viewer. *Nat. Biotechnol.* **29**, 3 (2011).

44. Ben-Kiki, O. & Evans, C. YAML Ain't Markup Language (YAMLTM) Version 1.2. 84.

45. Anders, S., Pyl, P. T. & Huber, W. HTSeq–a Python framework to work with high-throughput sequencing data. *Bioinformatics* **31**, 166–169 (2015).

46. McCarthy, D. J., Chen, Y. & Smyth, G. K. Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. *Nucleic Acids Res.* **40**, 4288–4297 (2012).

47. Karczewski, K. J. *et al. Variation across 141,456 human exomes and genomes reveals the spectrum of loss-of-function intolerance across human protein-coding genes*. http://biorxiv.org/lookup/doi/10.1101/531210 (2019) doi:10.1101/531210.

48. Amberger, J. S., Bocchini, C. A., Schiettecatte, F., Scott, A. F. & Hamosh, A. OMIM.org: Online Mendelian Inheritance in Man (OMIM®), an online catalog of human genes and genetic disorders. *Nucleic Acids Res.* **43**, D789–D798 (2015).

49. Richards, S. *et al.* Standards and guidelines for the interpretation of sequence variants: a joint consensus recommendation of the American College of Medical Genetics and Genomics and the Association for Molecular Pathology. *Genet. Med.* **17**, 405–423 (2015).

50. Lawrence, M. *et al.* Software for Computing and Annotating Genomic Ranges. *PLoS Comput. Biol.* **9**, e1003118 (2013).

51. Liao, Y., Smyth, G. K. & Shi, W. The R package Rsubread is easier, faster, cheaper and better for alignment and quantification of RNA sequencing reads. *Nucleic Acids Res.* **47**, e47 (2019).

52. Castel, S. E., Levy-Moonshine, A., Mohammadi, P., Banks, E. & Lappalainen, T. Tools and best practices for data processing in allelic expression analysis. *Genome Biol.* **16**, (2015).

53. Anders, S. & Huber, W. Differential expression analysis for sequence count data. *Genome Biol.* **11**, (2010).

54. Slatkin, M. Linkage disequilibrium — understanding the evolutionary past and mapping the medical future. *Nat. Rev. Genet.* **9**, 477–485 (2008).

55. Zheng, X. *et al.* A high-performance computing toolset for relatedness and principal component analysis of SNP data. *Bioinformatics* **28**, 3326–3328 (2012).

Acknowledgements

We thank Carla Andrade for helping with the DROP logo, as well as the members of the Gagneur Lab and Prokisch Lab for input. V.A.Y., C.M., H.P., and J.G. were supported by the EU Horizon2020 Collaborative Research Project SOUND (633974). The Bavaria California Technology Center supported C.M through a fellowship. The German Bundesministerium für Bildung und Forschung (BMBF) supported the study through the e:Med Networking fonds AbCD-Net (FKZ 01ZX1706A to V.A.Y., C.M., and J.G.), the German Network for Mitochondrial Disorders (mitoNET; 01GM1113C to H.P.), and the E-Rare project GENOMIT (01GM1207 to H.P.). A fellowship through the Graduate School of Quantitative Biosciences Munich supports V.A.Y.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- Fig1.png
- Table1.txt

- Fig3.png
- Table2.txt
- Fig2.png
- TableS1sampleannotation.txt
- DROPSupplementaryFigures.pdf
- Fig4.png
- Fig5.png