Supplemental Material for

VarFish - Comprehensive Variant Analysis for Diagnosis and Research

Manuel Holtgrewe[1,2,*], Oliver Stolpe[1,2], Mikko Nieminen[1,3], Stefan Mundlos[4,5], Alexej Knaus[6], Uwe Kornak[4,5], Dominik Seelow[4,7], Lara Segebrecht[4], Malte Spielmann[5,8], Björn Fischer-Zirnsak[4,5], Felix Boschann[4], Ute Scholl[9,7], Nadja Ehmke[4], Dieter Beule[1,3]

[1] CUBI – Core Unit Bioinformatics, Berlin Institute of Health, Berlin, 10117, Germany
[2] Charité – Universitätsmedizin Berlin, corporate member of Freie Universität Berlin, Humboldt-Universität zu Berlin, and Berlin Institute of Health, Berlin, 10117, Germany
[3] Max Delbrück Center for Molecular Medicine, Berlin, 13125, Germany
[4] Institute of Medical Genetics and Human Genetics, Charité – Universitätsmedizin Berlin, corporate member of Freie Universität Berlin, Humboldt-Universität zu Berlin, and Berlin Institute of Health, Berlin, 13353, Germany
[5] Development and Disease Group, Max Planck Institute for Medical Genetics, Berlin, 14195, Germany
[6] Institute for Genomic Statistics and Bioinformatics, University Hospital Bonn, Bonn, 53127, Germany
[7] Berlin Institute of Health (BIH), Anna-Louisa-Karsch-Str. 2, 10178 Berlin, Germany
[8] Institut für Humangenetik Lübeck, Universität zu Lübeck, 23538 Lübeck, Germany
[9] Charité – Universitätsmedizin Berlin, corporate member of Freie Universität Berlin, Humboldt-Universität zu Berlin, and Berlin Institute of Health, Department of Nephrology and Medical Intensive Care, BCRT – Berlin Institute of Health Center for Regenerative Therapies, 13353 Berlin, Germany

S1 In-House Database Feature in Variant Analysis

The following figures demonstrates how users can use the “in-house database” feature of VarFish.

For local (non-Kiosk mode) installations, VarFish computes statistics for each variant about the number of carries with heterozygous and homozygous state. Figure S1 shows how this can be used for filtering variants.

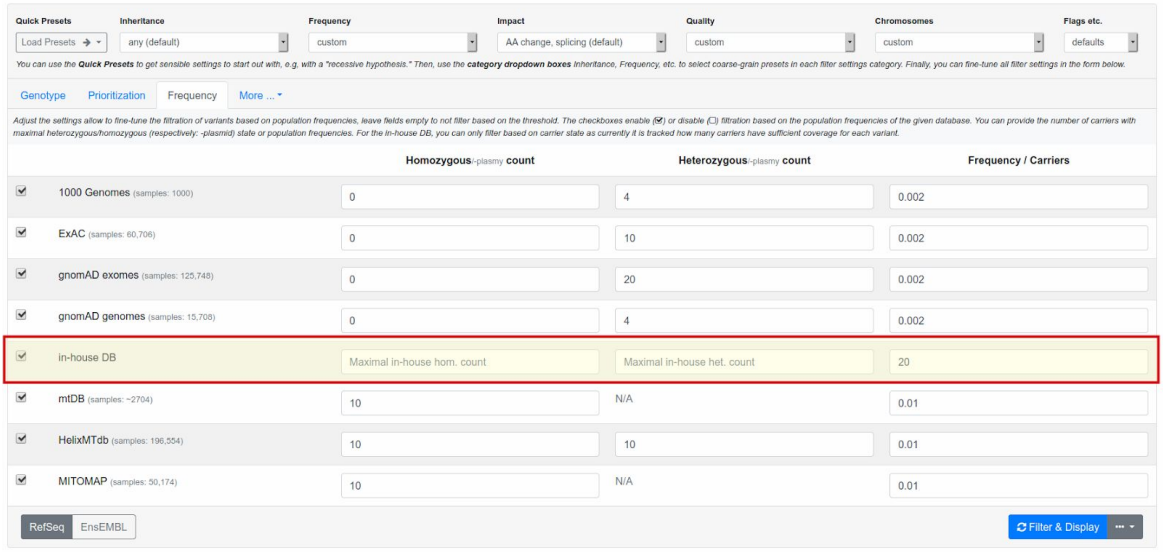
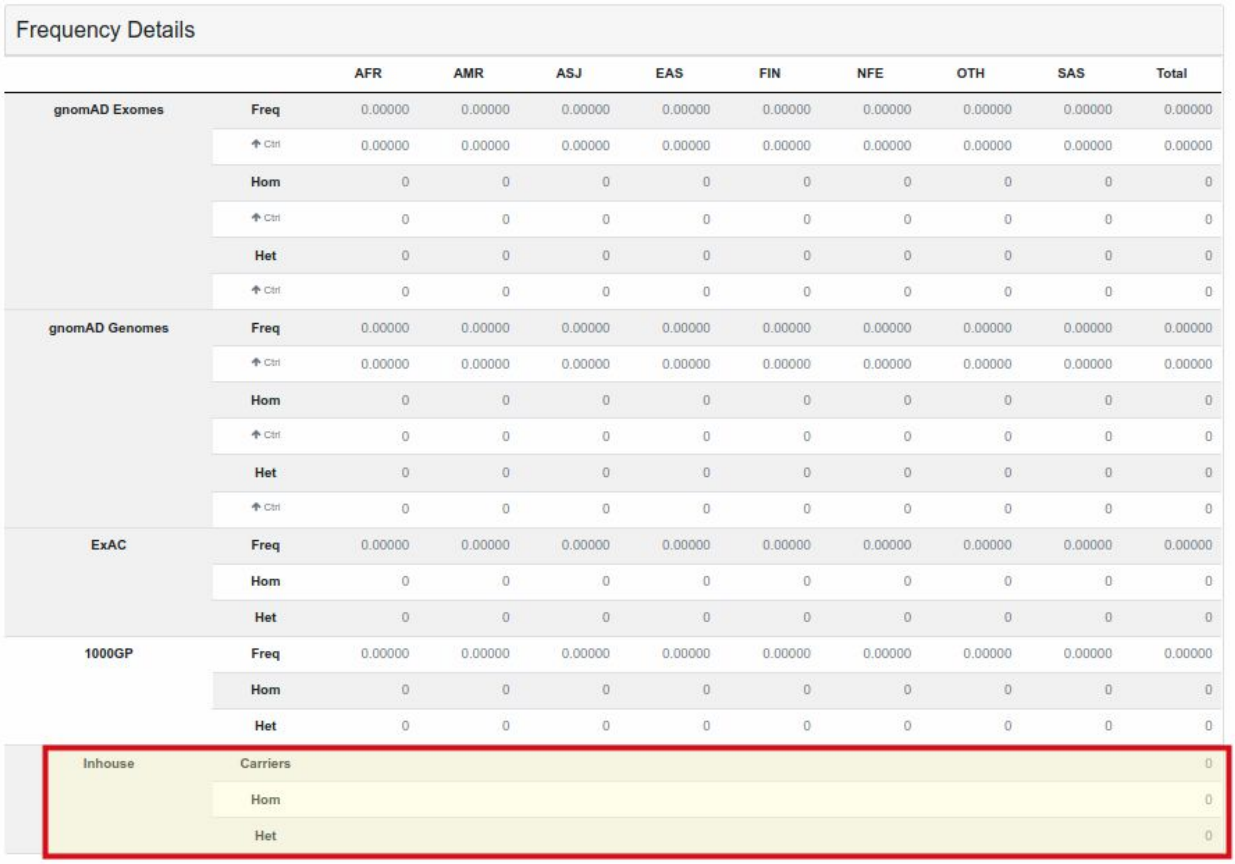


Figure S1. This figure shows the filter settings form for the “frequency” category. The row for adjusting the filter settings using the in-house database is highlighted. The user can filter variants based on their number of occurences in the in-house database in homozygous and heterozygous state or by the total number of carries. In the example above, variants with more than 20 carriers in the in-house database are removed.

For variants passing the frequency filter, the user might be interested in the number of total and homozygous carriers. This information is readily available in the result table (shown in Figure S2) after selecting “in-house DB” for the result frequency table (only frequencies from one database can be displayed in the overview at any given time).

Figure S2. This figure shows the in-house database frequency in the results table.

Finally, the in-house database frequencies are also available in the variant detail display (variant details are displayed when clicking the little angular bracket on the left of a variant result table row). This is shown in Figure S3.



Frequency Details

		AFR	AMR	ASJ	EAS	FIN	NFE	OTH	SAS	Total
gnomAD Exomes	Freq	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
	Ctrl	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
	Hom	0	0	0	0	0	0	0	0	0
	Ctrl	0	0	0	0	0	0	0	0	0
	Het	0	0	0	0	0	0	0	0	0
	Ctrl	0	0	0	0	0	0	0	0	0
gnomAD Genomes	Freq	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
	Ctrl	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
	Hom	0	0	0	0	0	0	0	0	0
	Ctrl	0	0	0	0	0	0	0	0	0
	Het	0	0	0	0	0	0	0	0	0
	Ctrl	0	0	0	0	0	0	0	0	0
ExAC	Freq	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
	Hom	0	0	0	0	0	0	0	0	0
	Het	0	0	0	0	0	0	0	0	0
1000GP	Freq	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
	Hom	0	0	0	0	0	0	0	0	0
	Het	0	0	0	0	0	0	0	0	0
Inhouse	Carriers									0
	Hom									0
	Het									0

Figure S3. This figure shows the variant frequency details table for the same variant as in Figure S2. The in-house database counts are shown in the same way as for the other population databases. Many columns remain empty because the in-house database does not have the population information available.

S2 User Annotation of Variants

In the results tables, user can open the “Flags & Comments” annotation window for a variant by clicking on the bookmark/bubble icon as show in Figure S4. The window is shown in Figure S5.

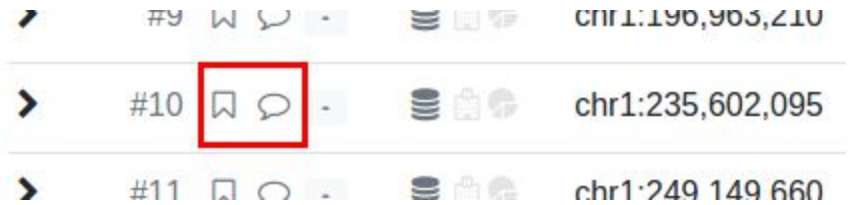


Figure S4. The bookmark/speech bubble triggers the “Flags & Comments” window shown in Figure S5.

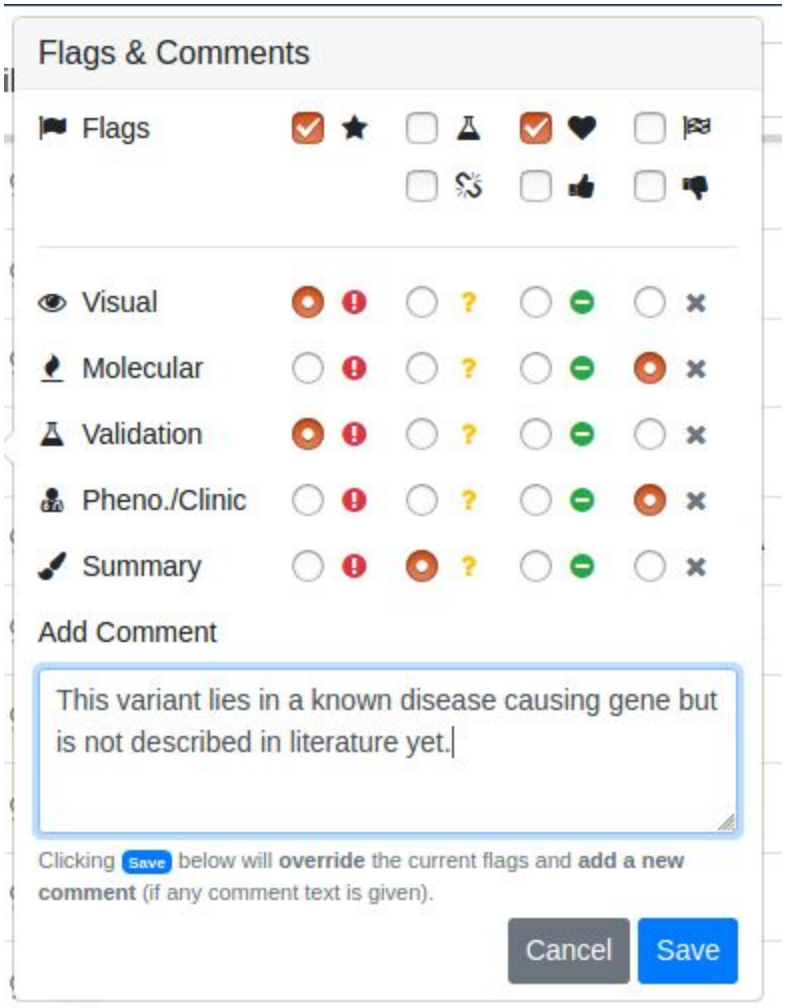


Figure S5. Users can assign flags and color ratings in different categories as well as text comments to variants.

The ACMG-AMP evaluation tool can be triggered by clicking on the current ACMG-AMP category display ("-" by default to indicate that no assessment has been performed yet) shown in Figure S6. The ACMG-AMP tool window is shown in Figure S7.

Figure S6. A click on the ACMG-AMP category display shows the ACMG-AMP tool shown in Figure S7.

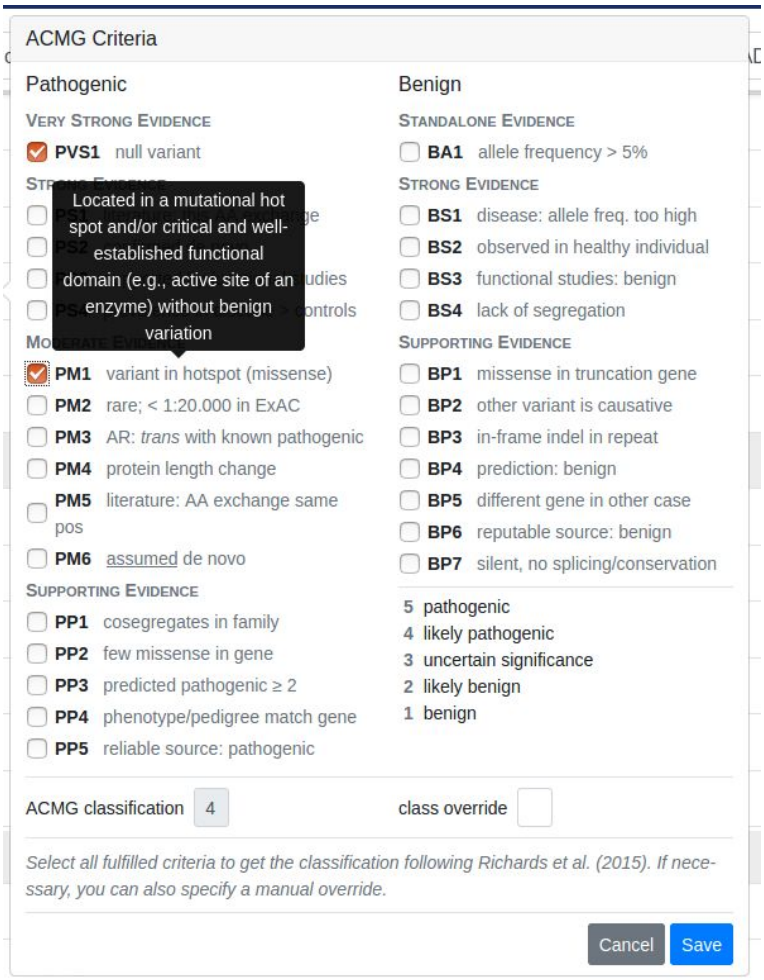


Figure S7. The ACMG-AMP tool window.

The result row for a variant indicates whether a variant has flags (filled bookmark symbol), comments (filled comments symbol), or ACMG-AMP ratings (colored number) is displayed in each result row as shown in figure S8.

Figure S8. A variant with bookmarks and comments (in red rectangle) and the ACMG-AMP assessment result (here “4” for “likely pathogenic”).

All annotations from the user are also displayed in the “Variant Annotation” tab of the case overview (as shown in Figure S9) and can also be listed for all cases in a project.

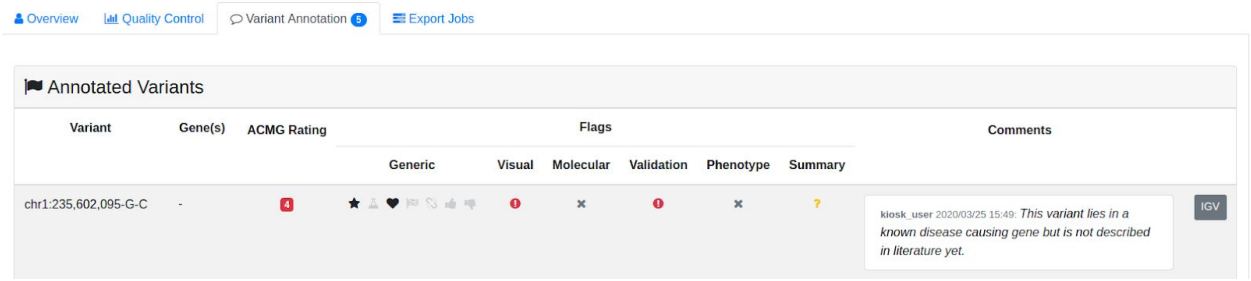


Figure S9. The variant annotation result display for the variant annotation illustrated in the figures of Section S2.

S3 VarFish SQL Query Generation

One aim in the development of VarFish is to allow for the interactive analysis of variants while at the same keeping all variants of an exome in the database, e.g., to allow for the in-house database feature. For the interactive usage, most queries must complete swiftly while keeping all variants means that tens of thousands of variants need to be kept. These two aims are somewhat conflicting as the processing time grows with the size of the processed data.

VarFish tackles this by employing three strategies: (1) using the star schema commonly found in data warehouse applications in combination with (2) indexes, and (3) data partitioning. We briefly explain each point.

1. All variants are stored in a central “variants” table with the basic information used for the filtration (including population frequencies, molecular impact, and genotypes in the user). All further annotation is stored in extra tables that can be joined with the central table in queries.
2. The VarFish database contains indices for the central variants table, one for each important class of queries. For example, many queries use the population frequencies for selecting rare variants. A database index targeting the frequency columns can be used for efficiently selecting a few hundred records of rare variants that are then processed further without index by the database server.
3. PostgreSQL also supports table partitioning. This allows to split a table by the numeric case ID. Each table partition can be considered independently which reduces the database index sizes and thus improves query performance.

While we have not performed any formal benchmarking, the strategy employed by VarFish is quite successful. For most use cases, users are interested in obtaining a short list of rare variants and potentially pathogenic variants. This list can be efficiently created by only considering variants with low population frequencies using the database index and then further filtering this shorter list.

Using the query generation approach from VarFish, the query execution is done by PostgreSQL which has an excellent query analyser and is able to perform the filtration efficiently. However, this approach also has the drawback that it is not possible to see how many variants passed which filter. First, VarFish only sends an SQL (standard query language) query to the database server and returns the final list of variants. Second, the database server will dynamically change the execution plan based on the query and the data itself (using internal counters and statistics). While it is possible to obtain the query execution plan of an executed query, it is infeasible to convert this into useful information for the VarFish user. Third, even if it was feasible to report it, the information would most probably not be useful to the user. The order of filter steps can be reordered by the PostgreSQL query optimizer when the user adjusts filter settings. Also, variants that do not pass a query criteria (e.g., population frequencies) are not further

considered (e.g., they are not filtered further for molecular impact). To summarize, using SQL query generation leads to very efficient data filtration at the cost of losing some transparency.