AutoACMG

Release 0.3.0

Dzmitry Hramyka

Sep 30, 2024

CONTENTS:

Auto-acmg is a comprehensive software suite designed for the automatic prediction of ACMG (American College of Medical Genetics and Genomics) criteria for the evaluation of sequence variants and copy number variations (CNVs) in genetic data. Its primary function is to assist geneticists and researchers by providing a systematic approach to classify genetic variants according to established guidelines. More information to the guidelines can be found at the *ACMG Sequence Variant Criteria* and *AMCG Sequence Variant Details* sections.

CHAPTER

ONE

KEY FEATURES

- **Variant Classification: Auto-acmg classifies single nucleotide variants (SNVs) and CNVs based** on ACMG guidelines, enhancing the accuracy and consistency of genetic interpretations.
- **Implementation of ACMG Criteria: It supports a range of criteria including PVS1, PS1, PM1,** PM2, PM4, PM5, PP2, PP3, BA1, BS1, BS2, BP1, BP3, BP4 and BP7, allowing users to assess the pathogenicity of variants in a comprehensive manner.
- **VCEP Specifications: The software is designed to adhere to Variant Curation Expert Panel** (VCEP) specifications, ensuring that each prediction is tailored to the specific genes and conditions studied by expert panels.
- **Extensible and Adaptable: Auto-acmg can be customized and extended to accommodate new criteria** and guidelines as genetic science evolves.

CHAPTER

TWO

GETTING STARTED

If you are new to auto-acmg, a good starting point is the *Usage* section which provides detailed instructions on how to use the software effectively.

If you are interested in contributing to auto-acmg, please refer to the *Dev Quickstart* section.

Also pay attention to the *AutoACMG Implementation* section to understand how auto-acmg works internally.

2.1 AutoACMG Implementation

AutoACMG operates through a structured process to determine the most appropriate predictor based on the type and context of the genetic variant being analyzed. This document describes the general workflow and the mechanisms behind the decisions that auto-acmg makes to predict the classification of genetic variants.

2.1.1 Workflow Overview

1. **Variant Resolution**: The first step involves determining the type of genetic variant. If the variant is recognized as a sequence variant (seqvar), auto-acmg employs the SeqVar default predictor. For structural variants (strucvar), the StrucVar default predictor is utilized, which primarily evaluates the PVS1 criteria.
2. **Gene-Specific Evaluation**: For seqvars, auto-acmg checks if the variant is associated with any gene that has a specific Variant Curation Expert Panel (VCEP) implementation. If a specific VCEP is applicable, auto-acmg will switch to the corresponding VCEP predictor for that gene to leverage expert-specified evaluation criteria and thresholds.
    - **SeqVar Implementation**: If no specific VCEP is applicable, the default SeqVar predictor is used. The implementation details of the default SeqVar predictor are described in *Implementation of different ACMG criteria for sequence variants*.
3. **Structural Variants**: For structural variants, the default predictor primarily evaluates the PVS1 criteria. You can find the implementation of the default StrucVar predictor in *Implementation of different ACMG criteria for structural variants*.

2.1.2 VCEP-Specific Overrides

Each VCEP-specific predictor may override certain logic or thresholds of the Default Predictor:

- **Threshold Adjustments: Some VCEP implementations may just adjust thresholds for criteria like** PP3 or BP4 and don't override the prediction logic.
- **Prediction Logic Changes: Other VCEPs may override the entire logic for certain criteria. This** can include changing how predictions are made based on the variant's effects on protein function, splicing, or other molecular mechanisms.

Note: For the **RYR1 gene**, there are two VCEPs: *Congenital Myopathies VCEP* and *Malignant Hyperthermia Susceptibility*. In AutoACMG, we only consider the **Malignant Hyperthermia Susceptibility VCEP** for RYR1 and avoid the Congenital Myopathies VCEP. For more details on the VCEP implementations, please refer to the source code in the `src/vcep` directory of our GitHub repository.

Note: For **von Willebrand Disease**, there are two different rulesets. We implement the combination of both rulesets. For more details on the VCEP implementations, please refer to the source code in the `src/vcep` directory of our GitHub repository.

For a detailed overview of how specific VCEP predictors modify or extend the default behavior, you can review the source code on the GitHub repository under `src/vcep` and for the details refer to the ClinGen VCEP Criteria Specification Registry.

2.1.3 Link to Source Code

Further details and the actual implementation of both default and VCEP-specific predictors can be accessed through our GitHub repository.

Note: The implementation specifics for each VCEP can vary significantly based on the gene and associated conditions. It is recommended to consult the individual VCEP documentation and the source code for precise information on how each gene-specific predictor is implemented.

2.2 AMCG Sequence Variant Details

This section describes additional details on our implementation of ACMG classification for sequence variants.

2.2.1 Data

The following datasources are used in the classification of sequence variants.

Table 1: Tools and datasources used

Data	Tool / Datasource	Original Datasource
Transcripts	Mehari / cdot	RefSeq / ENSEMBL
Variant Effect Predictions	Mehari	N/A
Variant Frequencies	annonars	gnomAD exomes, genomes, mtDNA
chrMT scores, annotations	annonars	MITOMAP, MitoTip, MitImpact, MtSNPscore
gene annotations	annonars	ClinGen, GCD, gene2phenotype, GenCC, PanelApp, DOMINO
protein annotations: domains, repeats, mutations	UCSC genome browser	UniProt

2.2.2 References

Literature with direct ACMG / ACGS / ClinGen relationship

- Richards S, Aziz N, Bale S, Bick D, Das S, Gastier-Foster J, Grody WW, Hegde M, Lyon E, Spector E, Voelkerding K, Rehm HL; ACMG Laboratory Quality Assurance Committee. *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. 2015 May;17(5):405-24. doi: 10.1038/gim.2015.30. Epub 2015 Mar 5. PMID: 25741868; PMCID: PMC4544753.
- ClinGen Sequence Variant Interpretation Work Group. *Recommendations for ACMG/AMP guideline criteria code modifications nomenclature.* November 10, 2017.
- Whiffin N, Minikel E, Walsh R, O'Donnell-Luria AH, Karczewski K, Ing AY, Barton PJR, Funke B, Cook SA, MacArthur D, Ware JS. *Using high-resolution variant frequencies to empower clinical genome interpretation.* Genet Med. 2017 Oct;19(10):1151-1158. doi: 10.1038/gim.2017.26. Epub 2017 May 18. PMID: 28518168; PMCID: PMC5563454.
- Abou Tayoun AN, Pesaran T, DiStefano MT, Oza A, Rehm HL, Biesecker LG, Harrison SM; ClinGen Sequence Variant Interpretation Working Group (ClinGen SVI). *Recommendations for interpreting the loss of function PVS1 ACMG/AMP variant criterion.* Hum Mutat. 2018 Nov;39(11):1517-1524. doi: 10.1002/humu.23626. Epub 2018 Sep 7. PMID: 30192042; PMCID: PMC6185798.
- Biesecker LG, Harrison SM; ClinGen Sequence Variant Interpretation Working Group. *The ACMG/AMP reputable source criteria for the interpretation of sequence variants.* Genet Med. 2018 Dec;20(12):1687-1688. doi: 10.1038/gim.2018.42. PMID: 29543229; PMCID: PMC6709533.
- Ghosh R, Harrison SM, Rehm HL, Plon SE, Biesecker LG; ClinGen Sequence Variant Interpretation Working Group. *Updated recommendation for the benign stand-alone ACMG/AMP criterion.* Hum Mutat. 2018 Nov;39(11):1525-1530. doi: 10.1002/humu.23642. PMID: 30311383; PMCID: PMC6188666.
- ClinGen Sequence Variant Interpretation Work Group. *SVI Recommendation for in trans Criterion (PM3) - Version 1.0* 2019.
- Brnich SE, Abou Tayoun AN, Couch FJ, Cutting GR, Greenblatt MS, Heinen CD, Kanavy DM, Luo X, McNulty SM, Starita LM, Tavtigian SV, Wright MW, Harrison SM, Biesecker LG, Berg JS; Clinical Genome Resource Sequence Variant Interpretation Working Group. *Recommendations for application of the functional evidence PS3/BS3 criterion using the ACMG/AMP sequence variant interpretation framework.* Genome Med. 2019 Dec 31;12(1):3. doi: 10.1186/s13073-019-0690-2. PMID: 31892348; PMCID: PMC6938631.
- Ellard S, Baple EL, Berry I, Forrester N, Turnbull C, Owens M, Eccles D, Abbs S, Scott R, Deans Z. *ACGS best practice guidelines for variant classification 2019.* 2019.
- ClinGen Sequence Variant Interpretation Work Group. *SVI Recommendation for Absence/Rarity (PM2) - Version 1.0* 2020.

- Ellard S, Baple EL, Callaway A, Berry I, Forrester N, Turnbull C, Owens M, Eccles DM, Abbs S, Scott R, Deans ZC, Lester T, Campbell J, Newman WG, Ramsden S, McMullan DJ. *ACGS Best Practice Guidelines for Variant Classification in Rare Disease 2020.* 2020.
- McCormick EM, Lott MT, Dulik MC, Shen L, Attimonelli M, Vitale O, Karaa A, Bai R, Pineda-Alvarez DE, Singh LN, Stanley CM, Wong S, Bhardwaj A, Merkurjev D, Mao R, Sondheimer N, Zhang S, Procaccio V, Wallace DC, Gai X, Falk MJ. *Specifications of the ACMG/AMP standards and guidelines for mitochondrial DNA variant interpretation.* Hum Mutat. 2020 Dec;41(12):2028-2057. doi: 10.1002/humu.24107. Epub 2020 Nov 10. PMID: 32906214; PMCID: PMC7717623.
- Tavtigian SV, Harrison SM, Boucher KM, Biesecker LG. *Fitting a naturally scaled point system to the ACMG/AMP variant classification guidelines.* Hum Mutat. 2020 Oct;41(10):1734-1737. doi: 10.1002/humu.24088. Epub 2020 Aug 30. PMID: 32720330; PMCID: PMC8011844.
- ClinGen Sequence Variant Interpretation Work Group. *SVI Recommendation for De Novo Criteria (PS2 & PM6) - Version 1.1* 2021.
- DiStefano MT, Goehringer S, Babb L, Alkuraya FS, Amberger J, Amin M, Austin-Tse C, Balzotti M, Berg JS, Birney E, Bocchini C. *The gene curation coalition: a global effort to harmonize gene-disease evidence resources.* Genetics in Medicine. 2022 Aug 1;24(8):1732-42.
- Pejaver V, Byrne AB, Feng BJ, Pagel KA, Mooney SD, Karchin R, O'Donnell-Luria A, Harrison SM, Tavtigian SV, Greenblatt MS, Biesecker LG, Radivojac P, Brenner SE; ClinGen Sequence Variant Interpretation Working Group. *Calibration of computational tools for missense variant pathogenicity classification and ClinGen recommendations for PP3/BP4 criteria.* Am J Hum Genet. 2022 Dec 1;109(12):2163-2177. doi: 10.1016/j.ajhg.2022.10.013. Epub 2022 Nov 21. PMID: 36413997; PMCID: PMC9748256.
- Walker LC, Hoya M, Wiggins GAR, Lindy A, Vincent LM, Parsons MT, Canson DM, Bis-Brewer D, Cass A, Tchourbanov A, Zimmermann H, Byrne AB, Pesaran T, Karam R, Harrison SM, Spurdle AB; ClinGen Sequence Variant Interpretation Working Group. *Using the ACMG/AMP framework to capture evidence related to predicted and observed impact on splicing: Recommendations from the ClinGen SVI Splicing Subgroup.* Am J Hum Genet. 2023 Jul 6;110(7):1046-1067. doi: 10.1016/j.ajhg.2023.06.002. Epub 2023 Jun 22. PMID: 37352859; PMCID: PMC10357475.

We currently exclude the following resources (we plan to later incorporate them):

- Variant Curation Expert Panel (VCEP) Criteria Specifications approved by the SVI VCEP Review Committee
- Strande NT, Riggs ER, Buchanan AH, Ceyhan-Birsoy O, DiStefano M, Dwight SS, Goldstein J, Ghosh R, Seifert BA, Sneddon TP, Wright MW, Milko LV, Cherry JM, Giovanni MA, Murray MF, O'Daniel JM, Ramos EM, Santani AB, Scott AF, Plon SE, Rehm HL, Martin CL, Berg JS. *Evaluating the Clinical Validity of Gene-Disease Associations: An Evidence-Based Framework Developed by the Clinical Genome Resource.* Am J Hum Genet. 2017 Jun 1;100(6):895-906. doi: 10.1016/j.ajhg.2017.04.015. Epub 2017 May 25. PMID: 28552198; PMCID: PMC5473734.

Further Supporting Literature

- Eilbeck K, Lewis SE, Mungall CJ, Yandell M, Stein L, Durbin R, Ashburner M. The Sequence Ontology: a tool for the unification of genome annotations. Genome Biol. 2005;6(5):R44. doi: 10.1186/gb-2005-6-5-r44. Epub 2005 Apr 29. PMID: 15892872; PMCID: PMC1175956.
- Quinodoz M, Royer-Bertrand B, Cisarova K, Di Gioia SA, Superti-Furga A, Rivolta C. *DOMINO: using machine learning to predict genes associated with dominant disorders.* The American Journal of Human Genetics. 2017 Oct 5;101(4):623-9.
- Kopanos C, Tsiolkas V, Kouris A, Chapple CE, Aguilera MA, Meyer R, Massouras A. *VarSome: the human genomic variant search engine.* Bioinformatics. 2019 Jun 6;35(11):1978.
- Martin AR, Williams E, Foulger RE, Leigh S, Daugherty LC, Niblock O, Leong IU, Smith KR, Gerasimenko O, Haraldsdottir E, Thomas E. *PanelApp crowdsources expert knowledge to establish consensus diagnostic gene panels.* Nature genetics. 2019 Nov;51(11):1560-5.

- Thormann A, Halachev M, McLaren W, Moore DJ, Svinti V, Campbell A, Kerr SM, Tischkowitz M, Hunt SE, Dunlop MG, Hurles ME. *Flexible and scalable diagnostic filtering of genomic variants using G2P with Ensembl VEP. Nature communications.* 2019 May 30;10(1):2373.
- Gudmundsson S, Singer-Berk M, Watts NA, Phu W, Goodrich JK, Solomonson M; Genome Aggregation Database Consortium; Rehm HL, MacArthur DG, O'Donnell-Luria A. *Variant interpretation using population databases: Lessons from gnomAD.* Hum Mutat. 2022 Aug;43(8):1012-1030. doi: 10.1002/humu.24309. Epub 2021 Dec 16. PMID: 34859531; PMCID: PMC9160216.

2.2.3 Criteria

The text in the following section is based on the one by Richards et al. (2015) and the updates listed in *References*.

Pathogenic Very Strong

PVS1 (null variant)

- variant is a null variant (sequence ontology: `stop_gained`, `frameshift_variant`, `splice_acceptor_variant`, `splice_donor_variant`, `start_lost`, `exon_loss_variant`, `transcript_variant`)
- loss of function is a known disease mechanism for the affected
- incorporate figures 4-5 from Walker et al. (2023)

Caveats

- beware of genes where LOF is not a known disease mechanism (e.g., GFAP, MYH7)
- caution when interpreting LOF at the extreme 3' and of gene
- caution with splice variants predicted to lead to exon skipping but leave the remainder of the protein intact
- caution in the presence of multiple transcripts

PVS1 Update 2018

Decision Tree

In Tayoun et al. (2018), the following decision tree is defined. It is based on the assumption that the gene-disease association is at a moderate, strong, or definitive clinical level according to Strande et al. (2017). Note that we do not incorporate the matching by Strande et al. (2017) for now.

1. **`stop_gained` or `frameshift_variant`**
    1. **predicted to undergo NMD**
        1. **exon is present in biologically-relevant transcripts**
            - result: `PVS1`
        2. **exon is absent from biologically-relevant transcripts**
            - result: N/A
    2. **not predicted to undergo NMD**
        1. **truncated / altered region is critical to protein function**
            - result: `PVS_Strong`

2. **role of region in protein function is unknown**
    1. **LoF variants in this exon are frequent in the general population and/or exon is absent from biologically-relevant transcripts**
        - result: N/A
    2. **LoF variants in this exon are not frequent in the general population and exon is present in biologically-relevant transcripts**

        variant removes >=10% of the protein

        - result: `PVS_Strong`

        variant removes <10% of the protein

        - result: `PVS1_Moderate`

2. `splice_acceptor_variant` or `splice_donor_variant`
    1. **exon skipping or use of a cryptic splice site disrupts reading frame and is predicted to undergo NMD**
        1. **exon is present in biologically-relevant transcripts**
            - result: `PVS1`
        2. **exon is absent from biologically-relevant transcripts**
            - result: N/A
    2. **exon skipping or use of a cryptic splice site disrupts reading frame and is NOT predicted to undergo NMD**
        1. **truncated / altered region is critical to protein function**
            - result: `PVS_Strong`
        2. **role of region in protein function is unknown**
            1. **LoF variants in this exon are frequent in the general population and/or exon is absent from biologically-relevant transcripts**
                - result: N/A
            2. **LoF variants in this exon are not frequent in the general population and exon is present in biologically-relevant transcripts**

                variant removes >=10% of the protein

                - result: `PVS_Strong`

                variant removes <10% of the protein

                - result: `PVS1_Moderate`
    3. **exon skipping or use of a cryptic splice site rpeserves reading frame**
        1. **role of region in protein is unknown**
            1. **LoF variants in this exon are frequent in the general population and/or exon is absent from biologically-relevant transcripts**
                - result: N/A
            2. **LoF variants in this exon are not frequent in the general population and exon is present in biologically-relevant transcripts**

                variant removes >=10% of the protein

- result: `PVS_Strong`

variant removes <10% of the protein

- result: `PVS1_Moderate`

2. **truncated / altered region is critical to protein function**

- result: `PVS_Strong`

3. `exon_loss_variant` or `transcript_variant` **(single exon or whole transcript deletion)**

1. **full gene deletion**

- result: `PVS1`

2. **single to multi exon deletion - disrupts reading frame and is predicted to undergo NMD**

1. **exon is present in biologically-relevant transcripts**

- result: `PVS1`

2. **exon is absent from biologically-relevant transcripts**

- result: N/A

3. **single to multi exon deletion - disrupts reading frame and is NOT predicted to undergo NMD**

1. **truncated/altered region is critical to protein function**

- result: `PVS_Strong`

2. **role of region in protein function is unknown**

1. **LoF variants in this exon are frequent in the general population and/or exon is absent from biologically-relevant transcripts**

- result: N/A

2. **LoF variants in this exon are not frequent in the general population and exon is present in biologically-relevant transcripts**

variant removes >=10% of the eprotein

- result: `PVS_Strong`

variant removes <10% of the protein

- result: `PVS1_Moderate`

4. **single to multi exon deletion - preserves reading frame**

1. role of region in protein function is unknown – see 3.3.2

2. **truncated / altered region is critical to protein function**

- result: `PVS_Strong`

4. **duplication (>=1 exon in size and must be contained within gene)**

1. **proven in tandem**

1. **reading frame disrupted and NMD predicted to occur**

- result: `PVS1`

2. **no or unknown impact on reading frame and NMD**

- result: N/!

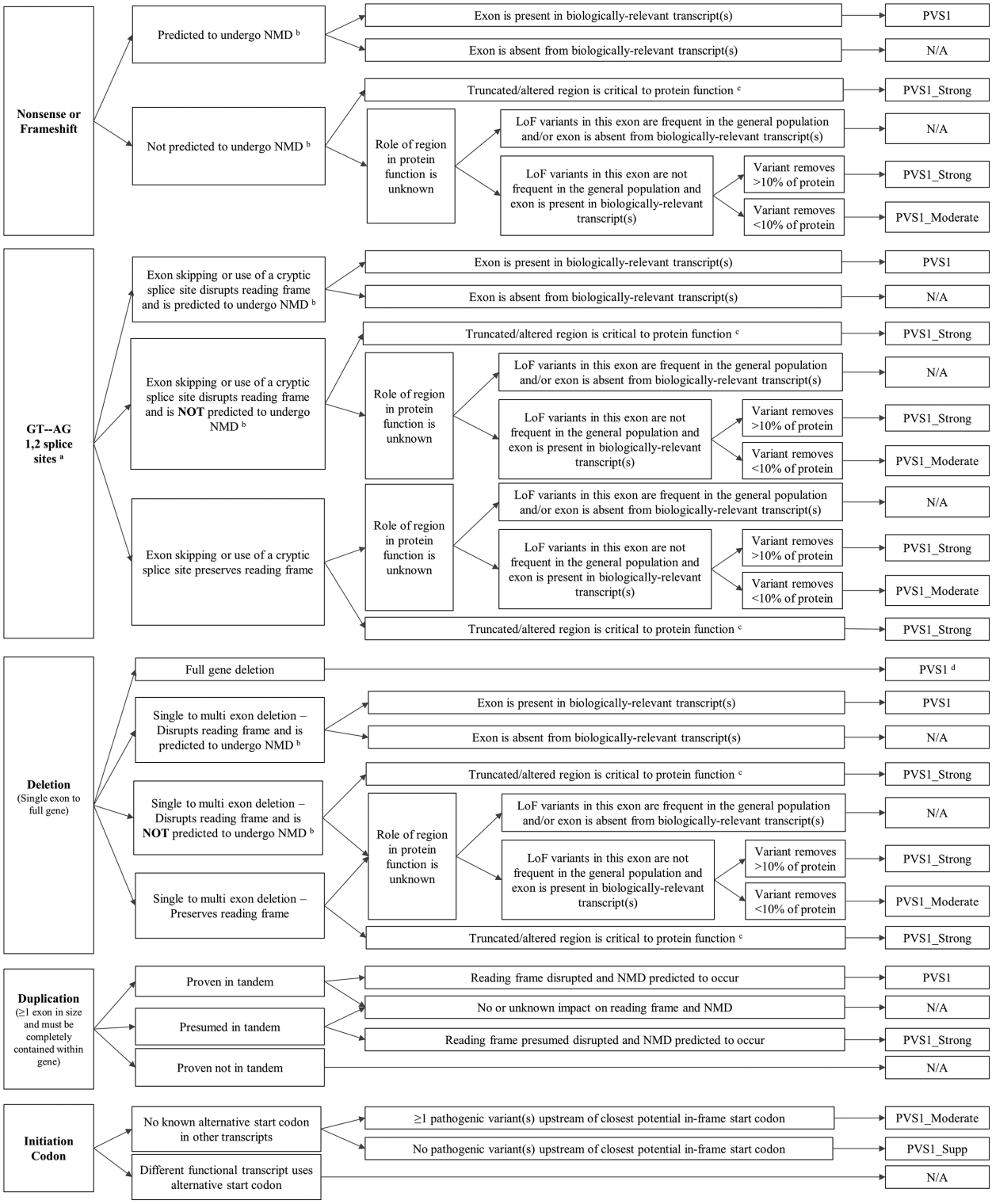
2. **presumed in tandem**

1. **reading frame presumed disrupted and NMD predicted to occur**
    - result: `PVS1`
2. **no or unknown impact on reading frame and NMD**
    - result: N/A

3. **proven not in tandem**
    - result: N/A

5. `start_lost`
    1. **no known alternative start codon in other transcripts**
        1. **>=1 pathogenic variant(s) upstream of closest potential in-frame start codon**
            - result: `PVS1_Moderate`
        2. **no pathogenic variant(s) upstream of closest potential in-frame start codon**
            - result: `PVS1_Supp`
    2. **different functional transcript uses alternative start codon**
        - result: N/A

And here is the tree as an image:



Notes

- criterion (2) `splice_acceptor_variant` or `splice_donor_variant` is mutually exclusive to splice site prediction
- "Generally, NMD is not predicted to occur if the premature termination codon occurs in the 3' most exon or within the 3' most 50 nucleotides of the penultimate exon" – from Tayoun et al. (2018)

Criteria for LoF Disease Mechanism

Further, Tayoun et al. (2018) define the following criteria for a loss-of-function disease mechanism.

1. **Follow PVS1 decision tree if:**
    - clinical validaity classification of gene is strong or definite AND
    - >=3 LoF functions are Pathogenic without PVS1 AND
    - >10% of variants associated with the phenotype are LoF (must be across more than 1 exon - except for single-exon genes)
2. **Decrease final strenght by one level (IOW: to `PVS1_Strong`) if:**
    - clinical validity classification of gene is at least moderate AND
    - >=2 LoF variants have previously associated with the phenotype (must be across more than one exon - except for single-exon genes) AND
    - null mouse model recapitulates disease phenotype
3. **Decrease final strength by two levels (IOW: to `PVS1_Moderate`) if:**
    - **clinical validity classification of gene is at least moderate AND EITHER**
        - >=2 LoF variants have been previously associated with the phenotype (must be across more than one exon - except for single-exon genes) OR
        - null mouse model recapitulates disease phenotype
4. If there is no evidence that LoF variants cause disease, PVS1 should not be applied at any strength level.

Pathogenic Strong

PS1 (same amino acid change)

- same amino acid change has previoulsy been established as pathogenic, regardless of nucleotide change
- for splicing variants, Tables 2-3 from Walker et al. (2023) shall be used

Table 2 Rules from Walker et al. (2023) of Variant under assessment (VUA)

- **VUA located outside splice donor / acceptor +/- 1/2 dinucleotide positions**
    - **baseline computational code: PP3**
        - **position of comparison variant relative to VUA: same nucleotide**
            - with P comparison variant: PS1
            - with LP comparison variant: PS1_Moderate
        - **position of comparison variant relative to VUA: within same splice donor / acceptor motif (including +/- 1/2 position)**
            - with P comparison variant: PS1_Moderate
            - with LP comparison variant: PS1_Supporting
- **VUA located at splice donor / acceptor +/- 1/2 dinucleotide positions**
    - **baseline computational code: PVS1**
        - position of comparison variant relative to VUA: within same splice donor / acceptor +/- 1/2 dinicleotide

        - with P comparison variant: VUA is PS1_Supporting
        - with LP comparison variant: N/A

    - **baseline computational code: PVS1**
        - position of comparison variant relative to VUA: within same splice donor / acceptor region but outside +/- 1/2 dinicleotide
        - with P comparison variant: VUA is PS1_Supporting
        - with LP comparison variant: VUA is PS1_Supporting

    - **baseline computational code: PVS1_Strong, PVS1_Moderate, PVS1_Supporting**
        - position of comparison variant relative to VUA: within same splice donor / acceptor +/- 1/2 dinicleotide
        - with P comparison variant: VUA is PS1
        - with LP comparison variant: VUA is N/A

    - **baseline computational code: PVS1_Strong, PVS1_Moderate, PVS1_Supporting**
        - position of comparison variant relative to VUA: within same splice donor / acceptor motify but outside +/- 1/2 dinicleotide
        - with P comparison variant: VUA is PS1_Moderate
        - with LP comparison variant: VUA is PS1_Supporting

Table 3. ACMG/AMP codes recommended for recording evidence relevant to variant position and predicted and experimentally observed impact on splicing

ACMG/AMP Code	Original definition[1]	(Re)definition for RNA impact	Notes
PVS1	PVS1. Null variant (nonsense, frameshift, "canonical ±1 or 2 splice sites," initiation codon, single or multi-exon deletion) in a gene where LoF is a known mechanism of disease.	bioinformatic data onry–variants impacting splice donor/ acceptor ±1,2 dinucleotides in a gene with established LoF as a disease mechanism	• develop gene-specific decision tree where feasible • use PVS1 decision tree to determine code strength
PVS1_Strength (RNA)	–	splicing assay data–assays demonstrating that a variant leads to aberrant splicing profile that can be categorized against a PVS1 decision tree	• PVS1_Strength (RNA) to be used to designate capture of splicing data (not PS3) • Use PVS1 decision tree to determine code strength. See Figure 5 flowchart. PVS1_Strength (RNA) may not be applicable for variants for which there is a plausible rescue model.
PS1	PS1. Same amino acid change as a previously established pathogenic variant regardless of nucleotide change	same predicted splicing impact as a previously classified (likely) pathogenic variant	• predicted event of variant matches that of a known (likely) pathogenic variant • Weights depend on relative positions of the variant under assessment and confidence in classification for the comparison variant. See Table 2.
PS3	PS3. Well-established *in vitro* or *in vivo* functional studies supportive of a damaging effect on the gene or gene product	not applicable for splicing effects	• replaced by PVS1_Strength (RNA), with weight determined as per PVS1 decision tree and other factors; see Figure 5 flowchart
PP3	PP3. Multiple lines of computational evidence support a deleterious effect on the gene or gene product (conservation, evolutionary, splicing impact, etc.). Caveat: Because many in *silico* algorithms use the same or very similar input for their predictions, each algorithm should not be counted as an independent criterion. PP3 can be used only once in any evaluation of a variant.	Computational evidence from calibrated prediction tool(s) supports impact on splicing.	• No requirement for multiple tools, but calibration of tool(s) to select cutoffs for predicting spliceogenicity is recommended (gene-specific calibration not required). • only use for variants located outside of the donor/ acceptor ±1,2 dinucleotide positions, when splicing assay data is not available • For exonic variants, also consider functional impact via encoded change in protein sequence.
BS3	BS3. Well-established *in vitro* or *in vivo* functional studies supportive of a damaging effect on the gene or gene product.	not applicable for splicing effects	• replaced by BP7_Strong (RNA); see Figure 5 flowchart.
BP4	BP4. Multiple lines of computational evidence suggest no impact on gene or gene product (conservation, evolutionary, splicing impact, etc.). Caveat: Because many in *silico* algorithms use the same or very similar input for their predictions, each algorithm cannot be counted as an independent criterion. BP4 can be used only once in any evaluation of a variant.	Computational evidence from calibrated prediction tool(s) supports no impact on splicing.	• No requirement for multiple tools, but calibration of tool(s) to select cutoffs for predicting spliceogenicity is recommended (gene-specific calibration not required). • May be applied in conjunction with BP7 and its derivatives for intronic/silent variants. See Figure 5 flowchart. • For exonic variants, also consider functional impact via encoded change in protein sequence.

(*Continued on next page*)

Table 3. *Continued*

ACMG/AMP Code	Original definition[3]	(Re)definition for RNA impact	Notes
BP7	BP7. A synonymous (silent) variant for which splicing prediction algorithms predict no impact to the "splice consensus sequence or the creation of a new splice site" AND the nucleotide is not highly conserved	synonymous (silent) variant or intronic variant with low prior probability of pathogenicity if no predicted impact on splicing	• Apply only if BP4 is met. • Evolutionary conservation is not considered informative for application of this code. • only applicable for intronic/non-coding variants outside the donor/acceptor splice region (conservatively designated as intronic variants at or beyond positions +7/−21) and synonymous (silent) exonic variants located outside of the first and the last 3 bases of the exon
BP7_Strong (RNA)	–	Splicing assay data demonstrating a variant is not associated with aberrantly spliced transcript(s) relative to transcript profiles in controls.	• BP7_Strong (RNA) to be used to designate capture of splicing data (not BS3) • See Figure 5 flowchart for guidance on weighting and combining with other codes. • For exonic variants, also consider functional impact via encoded change in protein sequence.

Table 2. PS1 code weights for variants with same predicted splicing event as a known (likely) pathogenic variant

Variant under assessment (VUA)	Baseline computational/ predictive code applicable to VUA	Position of comparison variant relative to VUA	PS1 code applicable to VUA	
			with P comparison variant	with LP comparison variant
Located outside splice donor/ acceptor ±1,2 dinucleotide positions	PP3	same nucleotide	PS1	PS1_Moderate
	PP3	within same splice donor/acceptor motif (including at ±1,2 positions)	PS1_Moderate	PS1_Supporting
Located at splice donor/acceptor ±1,2 dinucleotide positions	PVS1	within same splice donor/ acceptor ±1,2 dinucleotide	PS1_Supporting	N/A
	PVS1	within same splice donor/acceptor region, but outside ±1,2 dinucleotide[a]	PS1_Supporting	PS1_Supporting
	PVS1_Strong, PVS1_Moderate, or PVS1_Supporting	within same splice donor/ acceptor ±1,2 dinucleotide	PS1	N/A
	PVS1_Strong, PVS1_Moderate, or PVS1_Supporting	within same splice donor/acceptor motif, but outside ±1,2 dinucleotide[a]	PS1_Moderate	PS1_Supporting

Prerequisite for all: the predicted event of the VUA must precisely match the predicted event of the comparison (likely) pathogenic variant (e.g., both predicted to lead to exon skipping, or both to lead to enhanced use of a cryptic splice motif, AND the strength of the prediction for the VUA must be of similar or higher strength than the strength of the prediction for the comparison [likely] pathogenic variant). For an exonic variant, predicted or proven functional effect of missense substitution(s) encoded by the VUA and (likely) pathogenic variant should also be considered before application of this code. Dinucleotide positions refer to donor and acceptor dinucleotides in reference transcript(s) used for curation. Designated donor and acceptor motif ranges should be based on position weight matrices for intron category (see methods). For GT-AG introns these are defined as follows: the donor motif, last 3 bases of the exon and 6 nucleotides of intronic sequence adjacent to the exon; acceptor motif, first base of the exon and 20 nucleotides upstream from the exon boundary. Consider other motif ranges for non-GT-AG introns.

[a]If relevant, splicing assay data for a pathogenic variant outside a ±1,2 dinucleotide position may be used to update a PVS1 decision tree and hence the applicable PVS1 code for a ±1,2 dinucleotide variant.

Caveats

- beware of changes that impact splicing rather than amino acid or protein level

Notes

- incorporation of splicing here is based on the recommendation by Walker et al. (2023)

PS2 (confirmed *de novo*)

- confirmed *de novo* variant in a patient withou disease and no family history

Caveats

- confirmation of paternity only is insufficient (egg donation, surrogate motherhood, errors in embryo transfer, … can contribute to nonmaternity)

Notes

- ClinGen Sequence Variant Interpretation Work Group (2021) describe a point-scale for PS2 and PM6. However, this is hard to apply automatically as it requires an assessment of whether the phenotype is highly specific or consistent with the gene.

PS3 (functional studies)

- well-established in vitro or in vivo functional studies supportive of a damaging effect on the gene or gene product

Caveats

- functional studies that have been validated and shown to be reproducible and robust in a clinical diagnostic laboratory setting are considered the most well established

Notes

- There is further guidance in Brnich et al. (2020) on how to apply PS3 and BS3 when interpreting "well-established" functional assays.
- However, as this process is manual, it is not further considered here.
- Walker et al. (2023) is not considered here as the authors recommend to capture experimental evidence with PVS1 and is not suitable for automatic classification.

PS4 (prevalence)

- prevalence of the variant in affected individuals is significantly increased compared with the prevalence in controls

Caveats

- relative risk or OR, as obtained from case–control studies, is >5.0, and the confidence interval around the estimate of relative risk or OR does not include 1.0. See Richards et al. (2015) for detailed guidance.
- in instances of very rare variants where case–control studies may not reach statistical significance, the prior observation of the variant in multiple unrelated patients with the same phenotype, and its absence in controls, may be used as moderate level of evidence.

Pathogenic Moderate

PM1 (hotspot)

- located in a mutational hot spot and/or critical and well-established functional domain (e.g., active site of an enzyme) without benign variation

Caveats

- Pejaver et al. (2022) suggest to limit combined evidence from P1 and PP3 to strong

PM3 (recessive and in trans)

- for recessive disorders, detected in trans with a pathogenic or likely pathogenic variant in an affected patient

According to ClinGen Sequence Variant Interpretation Work Group (2019), there are points awarded per in *trans* proband (all variants should be sufficiently rare, thus meet PM specifiacation, P-Pathogenic or LP-Likely pathogenic):

Table 2: Points per proband

Classification / zygosity of other variant	Points per confirmed in *trans*	Points if phase unknown
Pathogenic or Likely pathogenic variant	1.0	0.5(P) or 0.25(LP)
Homozygous occurence (max point 1.0)	0.5	N/A
Uncertain significance variant	0.25	0.0

The resulting point rating gives the following evidence strength for PM3:

- 0.5-1.0: PM3_Supporting
- 1.0-2.0: PM3
- 2.0-4.0: PM3_Strong
- >=4.0: PM3_VeryStrong

Notes

- ClinGen Sequence Variant Interpretation Work Group (2019) changes this from "for recessive disorders, detected in trans with a pathogenic" to "for recessive disorders, detected in trans with a pathogenic or likely pathogenic variant in an affected patient"
- Further, this document introduces the point-based system from above.
- There are further considerations in ClinGen Sequence Variant Interpretation Work Group (2019) that are not considered here.

PM4 (protein length)

- protein length changes as a result of in-frame deletions/insertions in a nonrepeat region or stop-loss variants

PM5 (overlapping missense)

- novel missense change at an amino acid residue where a different missense change determined to be pathogenic has been seen before

Caveats

- beware of changes that impact splicing rather than at the amino acid/protein level.

PM6 (assumed *de novo*)

- assumed de novo, but without confirmation of paternity and maternity

Pathogenic Supporting

PM2_Supporting (absent from controls)

- absent from controls (or at extremely low frequency if recessive) in gnomAD

Notes

- population indel data is of high quality by now
- ClinGen Sequence Variant Interpretation Work Group (2020) has downgraded this to PM2_Supporting by default.

PP1 (cosegregaton)

- cosegregation with disease in multiple affected family members in a gene definitively known to cause th disease

Notes

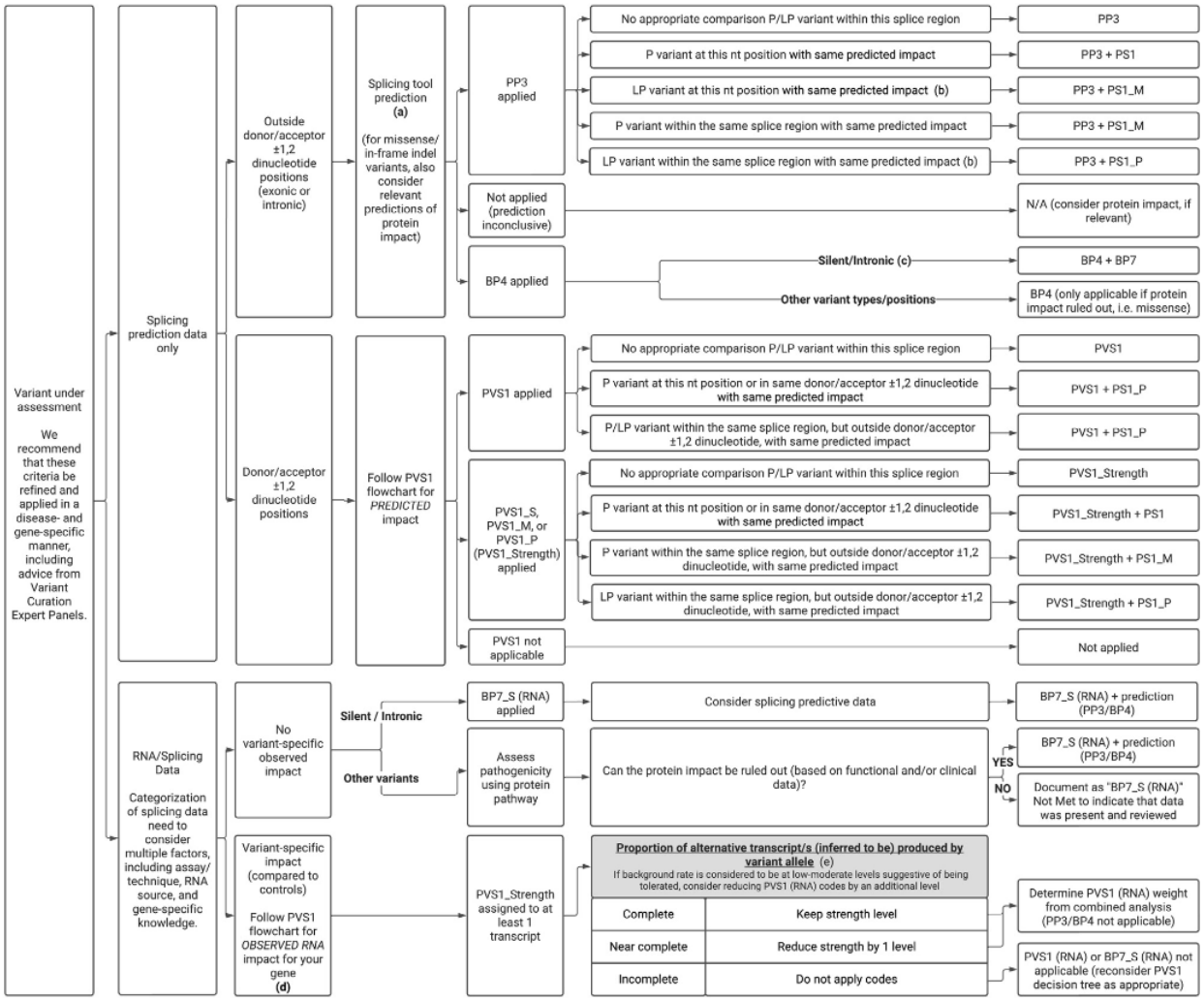
- may be used as stronger evidence with increasing segregation data

PP2 (missense)

- missense variant in a gene that has a low rate of benign missense variation and in which missense variants are a common mechanism of disease

PP3 (*in silico* predictions)

- multiple lines of computational evidence support a deleterious effect on the gene or gene product (conservation, evolutionary, splicing impact, etc.)
- incorporate figures 4-5 from Walker et al. (2023)



Caveats

- because many in silico algorithms use the same or very similar input for their predictions, each algorithm should not be counted as an independent criterion. PP3 can be used only once in any evaluation of a variant.
- Pejaver et al. (2022) suggest to limit combined evidence from P1 and PP3 to strong.

Notes

- The resulting class is updated according to the Pejaver et al. (2022). Note that it would be very useful to run the original code by Pejaver with more scores. The code from Pejaver can be found here on GitHub.

PP4 (monogenetic)

- patient's phenotype or family history is highly specific for a disease with a single genetic etiology

PP5 (reputable source)

Remove according to Biesecker et al. (2018).

Benign Standalone

BA1 (5% frequency)

- allele frequency is >5% in Exome Sequencing Project, 1000 Genomes Project, or Exome Aggregation Consortium

In accordance with Ghosh et al. (2018), there is a list of exceptions from this rule with high MAF but **some** evidence for pathogenicity. Updates to this list are available at ClinGen and shall be monitored regularly.

- `NM_014049.4(ACAD):c.-44_-41dupTAAG`
- `NM_004004.5(GJB2):c.109G>A (p.Val37Ile)`
- `NM_000410.3(HFE):c.187C>G (p.His63Asp)`
- `NM_000410.3(HFE):c.845G>A (p.Cys282Tyr)`
- `NM_000243.2(MEFV):c.1105C>T (p.Pro369Ser)`
- `NM_000243.2(MEFV):c.1223G>A (p.Arg408Gln)`
- `NM_006346.2(PIBF1):c.1214G>A (p.Arg405Gln)`
- `NM_000017.3(ACADS):c.511C>T (p.Arg171Trp)`
- `NM_000060.4(BTD):c.1330G>C (p.Asp444His)`

Benign Very Strong

This category does not exist in Richards et al. (2015) but is implicitely introduced by Tavtigian et al. (2020).

Benign Strong

BS1 (expected frequency)

- allele frequency is greater than expected for disorder

BS2 (healthy adult)

- observed in a healthy adult individual for a recessive (homozygous), dominant (heterozygous), or X-linked (hemizygous) disorder, with full penetrance expected at an early age

BS3 (functional studies)

- well-established in vitro or in vivo functional studies show no damaging effect on protein function or splicing

Notes

- There is further guidance in Brnich et al. (2020) on how to apply PS3 and BS3 when interpreting "well-established" functional assays. However, as this process is manual, it is not further considered here.
- Walker et al. (2023) is not considered here as the authors recommend to capture experimental evidence with PVS1 and is not suitable for automatic classification.

BS4 (lack of segregation)

- lack of segregation in affected members of a family

Caveat

- presence of phenocopies for common phenotypes (i.e., cancer, epilepsy) can mimic lack of segregation among affected individuals
- families may have more than one pathogenic variant contributing to an autosomal dominant disorder, further confounding an apparent lack of segregation

Benign Moderate

This category does not exist in Richards et al. (2015) but is implicitely introduced by Tavtigian et al. (2020).

Benign Supporting

BP1 (missense)

- missense variant in a gene for which primarily truncating variants are known to cause disease

BP2 (in trans)

- Observed in trans with a pathogenic variant for a fully penetrant dominant gene/disorder or observed in cis with a pathogenic variant in any inheritance pattern

BP3 (in-frame in repetitive)

- in-frame deletions/insertions in a repetitive region without a known function

BP4 (*in silico* predictions)

- multiple lines of computational evidence suggest no impact on gene or gene product (conservation, evolutionary, splicing impact, etc.)
- incorporate figures 4-5 from Walker et al. (2023)

Caveats

- because many in silico algorithms use the same or very similar input for their predictions, each algorithm cannot be counted as an independent criterion. BP4 can be used only once in any evaluation of a variant.

Notes

- The resulting class is updated according to the Pejaver et al. (2022). Note that it would be very useful to run the original code by Pejaver with more scores. The code from Pejaver can be found here on GitHub.

BP5 (found in solved)

- variant found in a case with an alternate molecular basis for disease

BP6 (reputable source)

Remove according to Biesecker et al. (2018).

BP7 (synonymous)

- synonymous (silent) variant for which splicing prediction algorithms predict no impact to the splice consensus sequence nor the creation of a new splice site AND the nucleotide is not highly conserved
- incorporate figures 4-5 from Walker et al. (2023)

2.3 ACMG Sequence Variant Criteria

This section describes the criteria as implemented for the automated classification of ACMG sequence variants.

A large focus is given towards the transparency to the user. The user is always given a report of the evidence for and against a given criterion. Even if not further described, the reason for skipping a criterion is always reported (e.g., BA1 disables all other benign criteria, or if a criterion is only valid for missense variants).

2.3.1 Mode of Inheritance

The mode of inheritance for a is derived from the following list of sources. The sources are iterated in the order given below and the first one with a match is used for deriving mode of inheritance.

1. **NHGRI CGD** The National Human Genome Research Institute Clinical Genomic Database
2. **ClinGen Disease Validity** Website
3. **EBI gene2phenotype** Website
4. **ClinGen GenCC** Website
5. **Genomics England PanelApp** Web-App
6. **Domino** is a machine learning method for prediction of mode of inheritance and described in PMID:28985496. We use the thresholds from PMID:30376034.
    - dominant: if score >= 0.5934
    - recessive: if score <= 0.3422
    - dominant/recessive: if score is between 0.3422 and 0.5934

Literature

Refer to *ACMG details* for the literature.

2.3.2 Allele Frequency

The databases gnomAD exomes and genomes will be used to derived allele frequencies. We trust the gnomAD quality control pipeline and filter solely above the gnomAD variant annotation. Frequencies are considered valid if:

- allele number is >=2000
- gnomAD quality filter is PASS

The criteria BA1 and BS1 will ignore the following "bottlenecked" populations: Amish (ami), Ashkenazi Jewish (asj), European Finnish (fin), Middle Eastern (mid), and "Remaining Individuals (rmi) groups".

2.3.3 Calibration

We currently do not have our own calibration yet. This is a big TODO once we have all the data.

2.3.4 Transcript Selection

We use the MANE Plus Clinical set of transcripts for the genes that have such information. Otherwise, we use the longest transcript as the MANE transcript and consider all alternate RefSeq transcripts.

MANE annotation is not available for GRCh37 and will not be provided by ENSEMBL/NCBI. We thus map the MANE (and MANE Plus Clinical) transcripts from GRCh38 by using the latest version of the transcript with the same identifier. If mapping either MANE or any of the MANE Plus Clinical transcripts fails then we fall back to the longest transcript rule.

2.3.5 Pathogenicity Predictions

We currently use the thresholds from Pejaver *et al.*[1] only.

Once we have our own calibration, we can extend our predictions to novel tools such as AlphaMissense.

2.3.6 Code Modification Nomenclature

In accordance with ClinGen Sequence Variant Interpretation Work Group (2017), modification codes are used. That is, for a criterion `${CRIT}`, the modification codes `${CRIT}_Supporting`, `${CRIT}_Moderate`, `${CRIT}_Strong`, `${CRIT}_VeryStrong`.

2.3.7 Rules / Point System

We consider three rule systems:

- The original ACMG 2015 rules
- The ACGS 2020 rules
- The 2020 Point system described by Tavtigian et al. (2020)

ACMG 2015 Rules

The following rules have been defined in Richards et al. (2015).

Pathogenic

If one of the following criteria 1-3 are fulfilled.

1. **1 very strong (PVS1) AND one of the following**
    a. >=1 strong (PS1-PS4)
    b. >=2 moderate (PM1-PM6)
    c. >=1 moderate (PM1-PM6) AND >=1 supporting (PP1-PP5)
    d. >=2 supporting (PP1-PP5)
2. >=2 strong (PS1-PS4)
3. **1 strong (PS1-PS4) AND**
    a. >=3 moderate (PM1-PM6)
    b. 2 moderate (PM1-PM6) AND >=2 supporting (PP1-PP5)
    c. 1 moderate (PM1-PM6) AND >=4 supporting (PP1-PP5)

[1] V. Pejaver, A. B. Byrne, B. J. Feng, K. A. Pagel, S. D. Mooney, R. Karchin, A. O'Donnell-Luria, S. M. Harrison, S. V. Tavtigian, M. S. Greenblatt, L. G. Biesecker, P. Radivojac, S. E. Brenner, L. G. Biesecker, S. M. Harrison, A. A. Tayoun, J. S. Berg, S. E. Brenner, G. R. Cutting, S. Ellard, M. S. Greenblatt, P. Kang, I. Karbassi, R. Karchin, J. Mester, A. O'Donnell-Luria, T. Pesaran, S. E. Plon, H. L. Rehm, N. T. Strande, S. V. Tavtigian, and S. Topper. Calibration of computational tools for missense variant pathogenicity classification and ClinGen recommendations for PP3/BP4 criteria. *Am J Hum Genet*, 109(12):2163–2177, Dec 2022.

Likely Pathogenic

If one of the following criteria 1-7 are fulfilled.

1. 1 very strong (PVS1) AND 1 moderate (PM1-PM6)
2. 1 strong (Ps1-PS4) AND 1-2 moderate (PM1-PM6)
3. 1 strong (PS1-PS4) AND >=2 supporting (PP1-PP5)
4. >=3 moderate (PM1-PM6)
5. 2 moderate (PM1-PM6) AND >=2 supporting (PP1-PP5)
6. 1 moderate (PM1-PM6) AND >=4 supporting (PP1-PP5)

Benign

If one of the following criteria 1-2 are fulfilled.

1. 1 standalone (BA1)
2. >=2 strong (BS1-BS4)

Likely Benign

If one of the following criteria 1-2 are fulfilled.

1. 1 strong (BS1-BS4) AND 1 supporting (BP1-BP7)
2. >=2 supporting (BP1-BP7)

Uncertain Significance

If if one of the following criteria 1-2 are fulfilled.

1. Other criteria shown above are not met
2. the criteria for benign and pathogenic are contradictory

ACGS 2020 Rules

The following is a refinement of the rules above set by the Ellard et al. (2020).

Pathogenic

1. **1 very strong (PVS) AND one of the following**
    a. >=1 strong
    b. >=1 moderate
    c. >=2 supporting
2. >=3 strong
3. **2 strong AND one of the following**

   a. >=1 moderate
   b. >=2 supporting
4. **1 strong AND one of the following**
   a. >=3 moderate
   b. >=2 moderate AND >=2 supporting
   c. >=1 moderate AND >=4 supporting

Likely Pathogenic

1. >=2 strong
2. **1 strong AND one of teh following**
   a. 1-2 moderate OR
   b. >=2 supporting
3. **>=3 moderate OR**
   a. 2 moderate AND >=2 supporting
   b. 1 modereate AND >=4 supporting

Tavtigian et al. (2020) Rules

Alternatively, Tavtigian et al. (2020) formulated the rules as an integer point system.

Table 2 from this manuscript gives point values:

Table 3: Points per proband

evidence strength	points pathogenic	points benign
indeterminate	0	0
supporting	1	-1
moderate	2	-2
strong	4	-4
very strong	8	-8

The point-based variant classification categories are then given in their Table 3:

Table 4: Categories and point ranges

category	point ranges
pathogenic	>= 10
likely pathogenic	6 to 9
uncertain significance	0 to 5
likely benign	-1 to -6
benign	<= -7

2.3.8 Criteria

PVS1 (null variant)

Original Definition

> Null variant (nonsense, frameshift, canonical +/-1 or 2 splice sites, initiation codon, single or multi-exon deletion) in a gene where loss of function (LOF) is a known mechanism of disease.
>
> Caveats:
>
> - Beware of genes where LOF is not a known disease mechanism (e.g. GFAP, MYH7)
> - Use caution interpreting LOF variants at the extreme 3' end of a gene
> - Use caution with splice variants that are predicted to lead to exon skipping but leave the remainder of the protein intact
> - Use caution in the presence of multiple transcripts
>
> —Richards et al. (2015); Table 4

Preconditions / Precomputations

- **Criterion establishes whether LoF is a known mechanism of disease:**
  - If at least 2 LoF variants are reported in ClinVar with two or more stars then this criterion is triggered.
  - If the gnomAD LOF Observed/Expected is less than 0.7555 then this criterion is triggered.
- **Criterion establishes whether a stop_gain variant introduced nonsense mediated decay (NMD) consistent with Abou Youn et al. (2018) and the VEP NMD plugin.**
  - If the variant is on chrMT then it cannot be NMD.
  - If the variant is not_stop gain then then it cannot be NMD, else:
  - If the variant is in the last exon of the transcript then it is predicted to escape NMD.
  - If the variant falls 50bp upstream of the penuultimate (second to the last) exon then it is predicted to escape NMD.
  - If the variant falls int the first 100 coding bases in teh transcript then it is predicted to escape NMD.
  - If the variant is in an intronless transcript, meaning only one exon exists in the transcript, then it is predicted to escape NMD.
  - Else, the variant is predicted to be NMD.
- The MANE Plus Clinical transcripts are used for "biologically relevant transcripts" in this criterion.

Implemented Criterion

While the original description is somewhat vague, the specification in Abou Tayoun et al. (2018) is more precise but complex to implement. We plan to implement it as closely as possible.

Refer to the dedicated section of PVS1 Algorithm for the algorithm.

Literature

- Richards et al. (2015) describes the original criterion.
- Abou Tayoun et al. (2018) describe refined criteria for PVS1.
- McCormick et al. (2020) describe the ACMG criteria for chrMT variants.
- **The following are from the VEP NMD plugin:**
  - Identifying Genes Whose Mutant Transcripts Cause Dominant Disease Traits by Potential Gain-of-Function Alleles (Coban-Akdemir, 2018)
  - The criteria and impact of nonsense-mediated mRNA decay in human cancers (Lindeboom, 2016)

User Report

The following information is reported to the user:

- The evidence for / against LoF as disease mechanism.
- Whether NMD and NMD escape is predicted for this variant and the reason.
- The use of MANE Plus Clinical or alternate transcripts for locating alternate start codons.
- Further information of interest from the Abou Tayoun et al. (2018) decision tree.

Caveats

- We use the thresholds from PMID:30376034 but should reconsider, e.g., switching to LOEUF here with our own thresholds.
- This is currently not implementing the full criteria set from Abou Tayoun et al. (2018).

Notes

- If this criterion is triggered then PP3 and PM4 will be disabled.

PS1 (same amino acid)

Original Definition

> Same amino acid change as a previously established pathogenic variant regardless of nucleotide change.
>
> Caveat: Beware of changes that impact splicing rather than at the amino acid/protein level.
>
> —Richards et al. (2015); Table 4

Preconditions / Precomputations

- If the variant is not a missense variant then this criterion is skipped.

Implemented Criterion

- Consider all equivalent missense variants in ClinVar.
- **If at least one of the variant then this criterion is triggered.**
  - If the variant has zero stars in ClinVar then we report PS1_Supporting only
  - If the variant has only one star in ClinVar then we report PS1_Moderate only
  - If the variant has two stars in ClinVar then we report PS1
  - If the variant has three stars or above in ClinVar then we report PS1_VeryStrong

User Report

- The selected variant in ClinVar and with assessment its star status with accession.
- All alternate variants in Clinvar with assessments and star status with accessions.

Literature

N/A

Caveats

- The wording of "established pathogenic" variant is not clear so we use the steps from above.
- Note that this also depends on disease match which the user must confirm manually.

PS2 (confirmed *de novo*)

No automation has been implemented.

Original Definition

> De novo (both maternity and paternity confirmed) in a patient with the disease and no family history
>
> Note: Confirmation of paternity only is insufficient. Egg donation, surrogate motherhood, errors in embryo transfer, etc. can contribute to non-maternity.
>
> —Richards et al. (2015); Table 4

PS3 (functional studies)

No automation has been implemented.

Original Definition

> Well-established in vitro or in vivo functional studies supportive of a damaging effect on the gene or gene product.
>
> Note: Functional studies that have been validated and shown to be reproducible and robust in a clinical diagnostic laboratory setting are considered the most well-established.
>
> —Richards et al. (2015); Table 4

PS4 (prevalence)

No automation has been implemented.

Original Definition

> The prevalence of the variant in affected individuals is significantly increased compared to the prevalence in controls
>
> Note 1: Relative risk (RR) or odds ratio (OR), as obtained from case-control studies, is >5.0 and the confidence interval around the estimate of RR or OR does not include 1.0. See manuscript for detailed guidance.
>
> > Note 2: In instances of very rare variants where case-control studies may not reach statistical significance, the prior observation of the variant in multiple unrelated patients with the same phenotype, and its absence in controls, may be used as moderate level of evidence.
> >
> > —Richards et al. (2015); Table 4

PM1 (hotspot)

Original Definition

> Located in a mutational hot spot and/or critical and well-established functional domain (e.g. active site of an enzyme) without benign variation.
>
> —Richards et al. (2015); Table 4

Preconditions / Precomputations

- If the variant is on chrMT then this criterion is skipped according to McCormick et al. (2020).

Implemented Criterion

- If the variant is within a hotspot (at least 4 pathogenic missense/in-frame variants within 25bp radius) then this criterion is triggered.
- If the variant is within an annotated UniProt domain and the domain contains at least 2 pathogenic variants then this criterion is triggered.

User Report

- The hotspot region definition and the number of pathogenic variants in the region.

Literature

- McCormick et al. (2020) describe the ACMG criteria for chrMT variants.

Caveats

- We currently use the threshold from PMID:30376034 and are lacking our own calibration.

PM3 (recessive in *trans*)

No automation has been implemented.

Original Definition

> For recessive disorders, detected in trans with a pathogenic variant.
>
> > Note: This requires testing of parents (or offspring) to determine phase.
>
> —Richards et al. (2015); Table 4

PM4 (protein length)

Original Definition

> Protein length changes due to in-frame deletions/insertions in a non-repeat region or stop-loss variants.
>
> —Richards et al. (2015); Table 4

Preconditions / Precomputations

- If PVS1 was triggered then this criterion is skipped to avoid double counting.
- If the variant is not an in-frame indel and not a stop-loss variant then this criterion is skipped.

Implemented Criterion

- **If the variant is an in-frame indel**
  - If the variant is inside a repeat masked region then it is skipped
  - If the variant is inside a repeat as annotated by UniProt then it is skipped
  - Otherwise, this criterion is triggered.
- If the variant is a stop-loss variant then this criterion is triggered.

User Report

- Any reasons for skipping in repeat regions.
- The transcript identifier.

Literature

N/A

Caveats

- Richards et al. (2015) state that the size of the indel and amount of change in amino acids should influence the classification. We currently do not have this implemented.

PM5 (overlapping missense)

Original Definition

> Novel missense change at an amino acid residue where a different missense change determined to be pathogenic has been seen before.
>
> Caveat: Beware of changes that impact splicing rather than at the amino acid/protein level.
>
> —Richards et al. (2015); Table 4

Preconditions / Precomputations

- **If the variant is on a nuclear chromosome**
  - If it is not a missense variant then this criterion is skipped.
- If the variant is on chrMT and not missense and not on a tRNA gene then this criterion is skipped.

Implemented Criterion

- **If the variant is on a nuclear chromosome:**
  - If the variant is at the same position as a pathogenic missense variant then this criterion is triggered.
- **If the variant is on chrMT:**
  - If the variant is a missense variant and at the same position as a pathogenic one then the criterion is triggered.
  - If the variant is on a tRNA gene and at the same position as a pathogenic one then the criterion is triggered as PM5_Supporting.

User Report

- The overlapping variant used for criterion.
- Any alternative overlapping variants not chosen.

Literature

- Richards et al. (2018) describes the criterion for nuclear chromosomes.
- McCormick et al. (2020) describes the criterion for chrMT.

Caveats

N/A

PM6 (assumed *de novo*)

No automation has been implemented.

Original Definition

> Assumed de novo, but without confirmation of paternity and maternity.
>
> —Richards et al. (2015); Table 4

PM2_Supporting (absent from controls)

Original Definition

> Absent from controls (or at extremely low frequency if recessive) in Exome Sequencing Project, 1000 Genomes or ExAC.
>
> —Richards et al. (2015); Table 4

Preconditions / Precomputations

- Determine *Mode of Inheritance* for the gene.
- Determine *Allele Frequency*.
- If the allele frequency is invalid then this criterion is skipped.

Implemented Criterion

- **If the variant is on a nuclear chromosome:**
  - **If the gene is marked as recessive or X-linked:**
    - If the variant allele count is <=4 then this criterion is triggered.
  - **If the gene is marked as dominant:**
    - If the homozygous allele count is <=1 then this criterion is triggered.
    - If the allele frequency is less than 0.0001 then this criterion is triggered.
- **If the variant is on chrMT:**
  If the variant frequency is below 0.00002=0.002%=1/50,000 then this criterion is triggered.

User Report

- The values and thresholds used by the criterion even if failed.

Literature

- Richards et al. (2015) describes the original criterion.
- ClinGen Sequence Variant Interpretation Work Group (2020): SVI Recommendation for Absence/Rarity (PM2) - Version 1.0 describes the downgrade to supporting.
- McCormick et al. (2020) describe the ACMG criteria for chrMT variants.

Caveats

- We currently use the threshold from PMID:30376034 and are lacking our own calibration.
- This criterion has been downgraded by default to supporting from strong in accordance to ClinGen Sequence Variant Interpretation Work Group (2020): *SVI Recommendation for Absence/Rarity (PM2) - Version 1.0*

PP1 (cosegregation)

No automation has been implemented.

PP2 (missense)

Original Definition

> Missense variant in a gene that has a low rate of benign missense variation and where missense variants are a common mechanism of disease.
>
> —Richards et al. (2015); Table 4

Preconditions / Precomputations

- If the variant is on chrMT then this criterion is skipped according to McCormick et al. (2020).
- If the variant is not a missense variant then this criterion is skipped.

Implemented Criterion

- If the ratio of pathogenic missense variants over all non-VUS missense variants is greater than 0.808 then this criterion is triggered.

User Report

- Report the ratio of pathogenic missense variants over all non-VUS missense variants.

Literature

- McCormick et al. (2020) describe the ACMG criteria for chrMT variants.

Caveats

- We currently use the threshold from PMID:30376034 and are lacking our own calibration.

Notes

- This criterion is similar to *BP1 (missense)*

PP3 (*in silico* predictions)

Original Definition

> Multiple lines of computational evidence support a deleterious effect on the gene or gene product (conservation, evolutionary, splicing impact, etc).
>
> Caveats:
>
> - As many in silico algorithms use the same or very similar input for their predictions, each algorithm should not be counted as an independent criterion.
> - PP3 can be used only once in any evaluation of a variant.
>
> —Richards et al. (2015); Table 4

Preconditions / Precomputations

- If the criterion PVS1 was triggered then this criterion is skipped.
- If the variant is on chrMT then it is skipped, as we don't have calibration for chrMT yet.
- If the variant is not found in dbNSFP or CADD precomputed scores then it is skipped as we don't have calibration for chrMT yet.

Implemented Criterion

An initial prediction is fist done using the general purpose pathogenicity predictors.

- **If we have a score from the following, then the prediction is used (in descending order of priority):**
  - REVEL, MutPred2, CADD, BayesDel, VEST4, …, PhyloP
  - we will use the modifiers from Pejaver *et al.*[Page 26, 1]
- If predictions are missing then then PhyloP of the position of the variant is used as a fallback.

Then, for splicing the following is done.

- If a SpliceAI prediction is performed then it is interpreted according to Walker *et al.*[2].

The highest-scoring variant is used for the final prediction.

[2] L. C. Walker, M. Hoya, G. A. R. Wiggins, A. Lindy, L. M. Vincent, M. T. Parsons, D. M. Canson, D. Bis-Brewer, A. Cass, A. Tchourbanov, H. Zimmermann, A. B. Byrne, T. Pesaran, R. Karam, S. M. Harrison, A. B. Spurdle, L. G. Biesecker, S. M. Harrison, A. A. Tayoun, J. S. Berg, S. E. Brenner, G. R. Cutting, S. Ellard, M. S. Greenblatt, P. Kang, I. Karbassi, R. Karchin, J. Mester, A. O'Donnell-Luria, T. Pesaran, S. E. Plon, H. L. Rehm, N. T. Strande, S. V. Tavtigian, and S. Topper. Using the ACMG/AMP framework to capture evidence related to predicted and observed impact on splicing: Recommendations from the ClinGen SVI Splicing Subgroup. *Am J Hum Genet*, 110(7):1046–1067, Jul 2023.

User Report

- The scores and predictions from the predictors.

Caveats

- As described in *Pathogenicity Predictions*, we are currently limited to the precomputed threshold from the literature. This hinders us in adopting AlphaMissense effectively, for example.
- We need to compute accuracy to rank the implemented methods.
- We need our own calibration for chrMT.

Notes

- This criterion is similar to *BP4 (in silico predictions)*

PP4 (monogenetic)

No automation has been implemented.

BA1 (5% frequency)

Original Definition

> Allele frequency is >5% in Exome Sequencing Project, 1000 Genomes Project, or Exome Aggregation Consortium
>
> —Richards et al. (2015); Table 4

Preconditions / Precomputations

- The variant is absent from the exception list from Ghosh et al. (2018). If the variant is present on this list, then this criterion is skipped.

Implemented Criterion

- **If the variant is nuclear (not on chrMT)**
  - If the allele frequency is above 0.05 in gnomAD global population then this criterion is triggered.
- **else (the variant is on chrMT)**
  - If the allele frequency is above 0.01 in gnomAD-mtDNA global population then this criterion is triggered.

User Report

- The variant frequency.

Literature

- Richards et al. (2015) describes the 5% allele frequency threshold.
- Ghosh et al. (2018) introduce the exception list and ClinGen maintains it.
- McCormick et al. (2020) describe the 1% allele frequency threshold as appropriate for chrMT variants.

Caveats

- The exception *"However, there must be no additional conflicting evidence to support pathogenicity, such as a novel occurrence in a certain haplogroup" from McCormick et al. (2020)* is not implemented yet.

BS1 (expected frequency)

Original Definition

> Allele frequency greater than expected for disorder.
>
> —Richards et al. (2015); Table 4

Preconditions / Precomputations

- Determine *Allele Frequency*.
- If the allele frequency is invalid then this criterion is skipped.

Implemented Criterion

- **If the variant is on a nuclear chromosome and the user provided a maximal credible population frequency:**
  - If the FAF from gnomAD is above the maximal credible population frequency then this criterion is triggered.
- **If the variant is on chrMT:**
  - If the population frequency is above 0.5% then this criterion is triggered in accordance to McCormick et al. (2020).

User Report

- The variant frequency and again the user specified maximal credible population frequency for nuclear variants.
- The variant frequency and the 0.5% threshold for chrMT variants.

Literature

- Richards et al. (2015) describes the original criterion without thresholds.
- Gudmundsson et al. (2022) describe the FAF threshold provided by gnomAD.
- McCormick et al. (2020) describe the ACMG criteria for chrMT variants.

BS2 (healthy adult)

Original Definition

> Observed in a healthy adult individual for a recessive (homozygous), dominant (heterozygous), or X-linked (hemizygous) disorder, with full penetrance expected at an early age.
>
> —Richards et al. (2015); Table 4

Preconditions / Precomputations

- If the criterion BA1 triggered then this criterion is skipped.
- Determine *Mode of Inheritance* for the gene.
- Determine *Allele Frequency*.
- If the allele frequency is invalid then this criterion is skipped.
- If the criterion BA1 was triggered then this criterion is skipped.

Implemented Criterion

- **If the gene is marked as recessive or X-linked:**
  - If the variant allele count is above 2 then this criterion is triggered.
- **If the gene is marked as dominant:**
  - If the variant allele count is above 5 then this criterion is triggered.

User Report

- The variant frequency and the threshold used.

Literature

- Chen et al. (2022), Karczewski et al. (2020), etc. describe gnomAD.
- The modes of inheritance for the genes are taken from different sources as described in *Mode of Inheritance*.

Caveats

- The conditions of "full penetrance" and "expected at an early age" need to be checked by the user.

Notes

- Genes can be marked as both recessive and dominant.
- We use the thresholds from PMID:30376034.

BS3 (functional studies)

No automation has been implemented.

Original Definition

> Well-established in vitro or in vivo functional studies shows no damaging effect on protein function or splicing.
>
> —Richards et al. (2015); Table 4

BS4 (lack of segregation)

No automation has been implemented.

Original Definition

> Lack of segregation in affected members of a family
>
> Caveats:
>
> - The presence of phenocopies for common phenotypes (i.e. cancer, epilepsy) can mimic lack of segregation among affected individuals.
> - Also, families may have more than one pathogenic variant contributing to an autosomal dominant disorder, further confounding an apparent lack of segregation.
>
> —Richards et al. (2015); Table 4

BP1 (missense)

Original Definition

> Missense variant in a gene for which primarily truncating variants are known to cause disease
>
> —Richards et al. (2015); Table 4

Preconditions / Precomputations

- If the criterion BA1 triggered then this criterion is skipped.
- If the variant is on chrMT then this criterion is skipped according to McCormick et al. (2020).
- If the variant is not a missense variant then this criterion is skipped.

Implemented Criterion

- If the ratio of benign missense variants over all non-VUS missense variants is greater than 0.569 then this criterion is triggered.

User Report

- Report the ratio of benign missense variants over all non-VUS missense variants together with threshold.

Literature

- McCormick et al. (2020) describe the ACMG criteria for chrMT variants.

Caveats

- We currently use the threshold from PMID:30376034 and are lacking our own calibration.

Notes

- This criterion is similar to *PP2 (missense)*

BP2 (recessive in *trans*)

No automation has been implemented.

Original Definition

> Observed in trans with a pathogenic variant for a fully penetrant dominant gene/disorder; or observed in cis with a pathogenic variant in any inheritance pattern
>
> —Richards et al. (2015); Table 4

BP3 (in-frame repetitive)

Note:

- We do not have proper Uniprot data yet (domain / repeat)
- Similar to repeat masker.
- Probably same for phylop100way?

Original Definition

> In-frame deletions/insertions in a repetitive region without a known function.
>
> —Richards et al. (2015); Table 4

Preconditions / Precomputations

- If the criterion BA1 triggered then this criterion is skipped.
- If the variant is on chrMT then this criterion is skipped.

Implemented Criterion

- If the variant is in a known functional domain according to UniProt then this criterion is skipped.
- If the variant is in a repeat region according to UniProt repeat annotation genome repeat masker then this criterion is skipped.
- If the variant is in a region of low conservation (PhyloP100Way less than 3.58, same as PMID:30376034) then this criterion is skipped.
- If all conditions above fail then this criterion is triggered.

User Report

- The variant position and the reason for triggering or skipping.

Literature

- McCormick et al. (2020) describe the ACMG criteria for chrMT variants.

Caveats

- We currently use the conservation threshold from PMID:30376034 and are lacking our own calibration.
- Different from PMID:30376034, we do not check whether there are known pathogenic variants in the region.

BP4 (*in silico* predictions)

Note:

- we have not implemented MitoTip or MitImpact yet
- we are lacking phylop scores yet
- we don't have live CADD scores yet

Original Definition

> Multiple lines of computational evidence suggest no impact on gene or gene product (conservation, evolutionary, splicing impact, etc).
>
> Caveat: As many in silico algorithms use the same or very similar input for their predictions, each algorithm cannot be counted as an independent criterion. BP4 can be used only once in any evaluation of a variant.
>
> —Richards et al. (2015); Table 4

Preconditions / Precomputations

- If the criterion BA1 triggered then this criterion is skipped.
- If the variant is on chrMT then it is skipped, as we don't have calibration for chrMT yet.
- If the variant is not found in dbNSFP or CADD precomputed scores then it is skipped as we don't have calibration for chrMT yet.

Implemented Criterion

See *PP3 (in silico predictions)* for details.

User Report

See *PP3 (in silico predictions)* for details.

Literature

See *PP3 (in silico predictions)* for details.

Caveats

See *PP3 (in silico predictions)* for details.

Notes

- This criterion is similar to *PP3 (in silico predictions)*

BP5 (found in solved)

No automation has been implemented.

Original Definition

> Variant found in a case with an alternate molecular basis for disease.
>
> —Richards et al. (2015); Table 4

BP7 (synonymous)

Original Definition

> A synonymous (silent) variant for which splicing prediction algorithms predict no impact to the splice consensus sequence nor the creation of a new splice site AND the nucleotide is not highly conserved.
>
> —Richards et al. (2015); Table 4

Preconditions / Precomputations

- If the variant is on chrMT then this criterion is skipped according to McCormick et al. (2020).

Implemented Criterion

- If there is a pathogenic variant +/- 2bp of the position in ClinVar then the criterion is skipped.
- If the variant is closer than 2bp to a splice site then the criterion is skipped.
- If the variant is not predicted to alter the splice site using SpliceAI then the criterion is triggered.

User Report

- The variant position and the reason for triggering or skipping.

Literature

- McCormick et al. (2020) describe the ACMG criteria for chrMT variants.

Caveats

N/A

Notes

- We use the thresholds from PMID:30376034.

Bibliography

2.4 Implementation of different ACMG criteria for sequence variants

This section contains the internal building blocks for the implementation of the different ACMG criteria for sequence variants.

2.4.1 PVS1

For sequence variants: PVS1 criteria for Sequence Variants (SeqVar).

class src.seqvar.auto_pvs1.**AutoPVS1**

Bases: *SeqVarPVS1Helper*

Handles the PVS1 criteria assessment for sequence variants.

_calc_alt_reg(*var_pos: int*, *exons: List[Exon]*, *strand: GenomicStrand*) → Tuple[int, int]

Calculates the altered region's start and end positions.

This method calculates the start and end positions of the altered region based on the position of the variant in the coding sequence and the exons of the gene. The method is implemented as follows: - If the genomic strand is plus, the start position is the variant position, and the end position is the last exon's end position. - If the genomic strand is minus, the start position is the first exon's start position, and the end position is the variant position.

Parameters

- **var_pos** – The position of the variant in the coding sequence.

- **exons** – A list of exons of the gene where the variant occurs.
- **strand** – The genomic strand of the gene.

Returns
The start and end positions of the altered region.

Return type
Tuple[int, int]

_closest_alt_start_cdn(*cds_info: Dict[str, CdsInfo]*, *hgvs: str*) → int | None

Calculate the closest potential start codon.

The method calculates the closest potential start codon based on the position of the variant in the coding sequence and the CDS information of the gene.

Parameters
- **cds_info** – A dictionary containing the CDS information for all transcripts.
- **hgvs** – The main transcript ID.

Returns
The position of the closest potential start codon, or None if not found.

Return type
Optional[int]

_convert_consequence(*var_data: AutoACMGSeqVarData*) → SeqVarPVS1Consequence

Convert the VEP consequence of the sequence variant to the internal representation.

Parameters
seqvar – The sequence variant being analyzed.

Returns
The internal representation of the VEP consequence.

Return type
SeqVarConsequence

_count_lof_vars(*seqvar: SeqVar*, *start_pos: int*, *end_pos: int*) → Tuple[int, int]

Counts Loss-of-Function (LoF) variants in the specified range.

The method retrieves variants from the specified range and iterates through the available data of each variant. The method counts the number of LoF variants and the number of frequent LoF variants in the specified range, based on the gnomAD genomes data (for the consequence of Nonsense and Frameshift variants) and the allele frequency (for the frequency of the LoF variants in the general population).

Note: A LoF variant is considered frequent if its occurrence in the general population exceeds some threshold. We use a threshold of 0.1% to determine if the LoF variant is frequent.

Parameters
- **seqvar** – The sequence variant being analyzed.
- **start_pos** – The start position of the range.
- **end_pos** – The end position of the range.

Returns
The number of frequent LoF variants and the total number of LoF variants.

Return type
Tuple[int, int]

_count_pathogenic_vars(*seqvar: SeqVar*, *start_pos: int*, *end_pos: int*) → Tuple[int, int]

Counts pathogenic variants in the specified range.

The method retrieves variants from the specified range and iterates through the ClinVar data of each variant to count the number of pathogenic variants and the total number of variants.

Parameters
- **seqvar** – The sequence variant being analyzed.

- **start_pos** – The start position of the range.
- **end_pos** – The end position of the range.

Returns
The number of pathogenic variants and the total number of variants.

Return type
Tuple[int, int]

Raises
InvalidAPIResposeError – If the API response is invalid or cannot be processed.

_find_aff_exon_pos(*var_pos: int*, *exons: List[Exon]*) → Tuple[int, int]

Find start and end positions of the affected exon.

Parameters
- **var_pos** – The position of the variant in the coding sequence.
- **exons** – A list of exons of the gene where the variant occurs.

Returns
The start and end positions of the affected exon.

Return type
Tuple[int, int]

Raises
AlgorithmError – If the affected exon is not found.

_get_conseq(*val: SeqVarPVS1Consequence*) → List[str]

Get the VEP consequence of the sequence variant by value.

Parameters
val – The value of the consequence.

Returns
The VEP consequences of the sequence variant.

Return type
List[str]

_skipping_exon_pos(*seqvar: SeqVar*, *exons: List[Exon]*) → Tuple[int, int]

Calculate the length of the closest to the seqvar exon.

The method calculates the length of the exon, which can be skipped due to the variant consequences.

Parameters
- **seqvar** – The sequence variant being analyzed.
- **exons** – A list of exons of the gene where the variant occurs.

Returns
The start and end positions of the exon skipping region.

Return type
Tuple[int, int]

alt_start_cdn(*cds_info: Dict[str, CdsInfo]*, *hgvs: str*) → bool

Check if the variant introduces an alternative start codon in other transcripts.

Implementation of the rule:
- **Iterating through all transcripts and checking if the coding sequence start** differs from the main transcript.
- If the start codon differs, the rule is met.

Note:

Rule:
If the variant introduces an alternative start codon in other transcripts, it is considered to be non-pathogenic.

Parameters

- **hgvs** – The main transcript ID.
- **cds_info** – A dictionary containing the CDS information for all transcripts.

Returns

True if the variant introduces an alternative start codon in other transcripts, False otherwise.

Return type

bool

annonars_client

Annonars client for the API.

comment_pvs1: str

Comment to store the prediction explanation.

crit4prot_func(*seqvar: SeqVar*, *exons: List[Exon]*, *strand: GenomicStrand*) → bool

Checks if the truncated or altered region is critical for the protein function.

This method assesses the impact of a sequence variant based on the presence of pathogenic variants downstream of the new stop codon, utilizing both experimental and clinical evidence.

Implementation of the rule: - Calculating the range of the altered region, based on the position of the variant in the coding sequence and the exons of the gene. - Fetching variants from the specified range of the altered region. - Counting the number of pathogenic variants in that region, by iterating through the clinvar data of each variant. - Considering the region critical if the frequency of pathogenic variants exceeds 5%.

Note: The significance of a truncated or altered region is determined by the presence and frequency of pathogenic variants downstream from the new stop codon. We use a threshold of 5% to determine if the region is critical for the protein function.

Parameters

- **seqvar** – The sequence variant being analyzed.
- **cds_pos** – The position of the variant in the coding sequence.
- **exons** – A list of exons of the gene where the variant occurs.
- **strand** – The genomic strand of the gene.

Returns

True if the altered region is critical for the protein function, otherwise False.

Return type

bool

Raises

InvalidAPIResponseError – If the API response is invalid or cannot be processed.

exon_skip_or_cryptic_ss_disrupt(*seqvar: SeqVar*, *exons: List[Exon]*, *consequences: List[str]*, *strand: GenomicStrand*) → bool

Check if the variant causes exon skipping or cryptic splice site disruption.

The method checks if the variant causes exon skipping or cryptic splice site disruption based on the position of the variant in the coding sequence and the exons of the gene.

Implementation of the rule: - If the exon length is not a multiple of 3, the variant is predicted to cause exon skipping. - If the variant is a splice acceptor or donor variant, the method predicts cryptic splice site disruption.

Note: Rule: If the variant causes exon skipping or cryptic splice site disruption, it is considered to be pathogenic.

Parameters

- **seqvar** – The sequence variant being analyzed.
- **exons** – A list of exons of the gene where the variant occurs.
- **consequences** – A list of VEP consequences of the sequence variant.

Returns

True if the variant causes exon skipping or cryptic splice site disruption, False if preserves reading frame.

Return type

bool

in_bio_relevant_tx(*transcript_tags: List[str]*) → bool

Checks if the exon with SeqVar is in a biologically relevant transcript.

Implementation of the rule:

If the variant is located in a transcript with a MANE Select tag, it is considered to be in a biologically relevant transcript.

Parameters

transcript_tags – A list of tags for the transcript.

Returns

True if the variant is in a biologically relevant transcript, False otherwise.

Return type

bool

lof_freq_in_pop(*seqvar: SeqVar*, *exons: List[Exon]*, *strand: GenomicStrand*) → bool

Checks if the Loss-of-Function (LoF) variants in the exon are frequent in the general population.

This function determines the frequency of LoF variants within a specified genomic region and evaluates whether this frequency exceeds a defined threshold indicative of common occurrence in the general population.

Implementation of the rule: - Calculating the range of the altered region (coding sequence of the transcript). - Counting the number of LoF variants and frequent LoF variants in that region. - Considering the LoF variants frequent in the general population if the frequency of "frequent" LoF variants exceeds 10%.

Note: A LoF variant is considered frequent if its occurrence in the general population exceeds some threshold. We use a threshold of 10% to determine if the LoF variant is frequent.

Parameters

- **seqvar** – The sequence variant being analyzed.
- **exons** – A list of exons of the gene where the variant occurs.
- **strand** – The genomic strand of the gene.

Returns

True if the LoF variant frequency is greater than 10%, False otherwise.

Return type

bool

Raises

InvalidAPIResponseError – If the API response is invalid or cannot be processed.

lof_rm_gt_10pct_of_prot(*prot_pos: int*, *prot_length: int*) → bool

Check if the LoF variant removes more than 10% of the protein.

The method checks if the LoF variant removes more than 10% of the protein based on the position of the variant in the protein and the length of the protein.

Note:

Rule:

A LoF variant is considered to remove more than 10% of the protein if the variant removes more than 10% of the protein.

Parameters
- **prot_pos** – The position of the variant in the protein.
- **prot_length** – The length of the protein.

Returns
True if the LoF variant removes more than 10% of the protein, False otherwise.

Return type
bool

predict_pvs1(*seqvar: SeqVar*, *var_data: AutoACMGSeqVarData*) → AutoACMGCriteria

Predict the PVS1 criteria.

undergo_nmd(*var_pos: int*, *hgnc_id: str*, *strand: GenomicStrand*, *exons: List[Exon]*) → bool

Classifies if the variant undergoes Nonsense-mediated decay (NMD).

Implementation of the rule: The method checks if the variant is in the GJB2 gene and always predicts it to undergo NMD. If the variant is not in the GJB2 gene, the method checks if the new stop codon position including the 5' UTR length is less or equal to the NMD cutoff, and if it is indeed less or equal, the variant is predicted to undergo NMD. Otherwise, the variant is predicted to escape NMD.

Note:

Rule:
If the variant is located in the last exon or in the last 50 nucleotides of the penultimate exon, it is NOT predicted to undergo NMD.

Important:
For the GJB2 gene (HGNC:4284), the variant is always predicted to undergo NMD.

Parameters
- **var_pos** – The position of the new stop codon !including the 5' UTR length!.
- **hgnc_id** – The HGNC ID of the gene.
- **strand** – The genomic strand of the gene.
- **exons** – A list of exons of the gene.

Returns
True if the variant undergoes NMD, False if variant escapes NMD.

Return type
bool

up_pathogenic_vars(*seqvar: SeqVar*, *exons: List[Exon]*, *strand: GenomicStrand*) → bool

Look for pathogenic variants upstream of the closest potential in-frame start codon.

The method checks for pathogenic variants upstream of the closest potential in-frame start codon. The method is implemented as follows: - Find the closest potential in-frame start codon. - Fetch and count pathogenic variants in the specified range. - Return True if pathogenic variants are found, otherwise False.

Parameters
- **seqvar** – The sequence variant being analyzed.
- **cds_pos** – The position of the variant in the coding sequence.
- **exons** – A list of exons of the gene where the variant occurs.
- **strand** – The genomic strand of the gene.

Returns
True if pathogenic variants are found upstream of the closest potential in-frame start codon, False otherwise.

Return type
bool

verify_pvs1(*seqvar: SeqVar*, *var_data: AutoACMGSeqVarData*) → Tuple[PVS1Prediction, PVS1PredictionSeqVarPath, str]

Make the PVS1 prediction.

The prediction is based on the PVS1 decision tree for sequence variants. The prediction and prediction path is stored in the prediction and prediction_path attributes.

Parameters

- **seqvar** – The sequence variant being analyzed.
- **var_data** – The data of the sequence variant.

Returns

The prediction, prediction path,
and the comment.

Return type

Tuple[PVS1Prediction, PVS1PredictionSeqVarPath, str]

src.seqvar.auto_pvs1.**CUSTOM_VCEP_PVS1 = ['HGNC:92', 'HGNC:7551', 'HGNC:1748', 'HGNC:4175', 'HGNC:4136', 'HGNC:3546', 'HGNC:3551', 'HGNC:7720', 'HGNC:129', 'HGNC:7448', 'HGNC:10483', 'HGNC:17098', 'HGNC:1100', 'HGNC:1101', 'HGNC:10585', 'HGNC:10588', 'HGNC:10590', 'HGNC:10596', 'HGNC:10586', 'HGNC:6547', 'HGNC:795', 'HGNC:26144', 'HGNC:175', 'HGNC:3349', 'HGNC:583', 'HGNC:7127', 'HGNC:7325', 'HGNC:7329', 'HGNC:9122', 'HGNC:10294', 'HGNC:4065', 'HGNC:11621', 'HGNC:5024', 'HGNC:4195', 'HGNC:10471', 'HGNC:8582', 'HGNC:9588', 'HGNC:6742', 'HGNC:11634', 'HGNC:11079', 'HGNC:11411', 'HGNC:3811', 'HGNC:6990', 'HGNC:12496', 'HGNC:12765', 'HGNC:186', 'HGNC:17642', 'HGNC:6024', 'HGNC:6193', 'HGNC:9831', 'HGNC:9832', 'HGNC:6010', 'HGNC:11998', 'HGNC:12687']**

List of genes with custom PVS1 criteria.

class src.seqvar.auto_pvs1.**SeqVarPVS1Helper**

Bases: AutoACMGHelper

Helper methods for PVS1 criteria for sequence variants.

_calc_alt_reg(*var_pos: int*, *exons: List[Exon]*, *strand: GenomicStrand*) → Tuple[int, int]

Calculates the altered region's start and end positions.

This method calculates the start and end positions of the altered region based on the position of the variant in the coding sequence and the exons of the gene. The method is implemented as follows: - If the genomic strand is plus, the start position is the variant position, and the end position is the last exon's end position. - If the genomic strand is minus, the start position is the first exon's start position, and the end position is the variant position.

Parameters

- **var_pos** – The position of the variant in the coding sequence.
- **exons** – A list of exons of the gene where the variant occurs.
- **strand** – The genomic strand of the gene.

Returns

The start and end positions of the altered region.

Return type

Tuple[int, int]

_closest_alt_start_cdn(*cds_info: Dict[str, CdsInfo]*, *hgvs: str*) → int | None

Calculate the closest potential start codon.

The method calculates the closest potential start codon based on the position of the variant in the coding sequence and the CDS information of the gene.

Parameters

- **cds_info** – A dictionary containing the CDS information for all transcripts.
- **hgvs** – The main transcript ID.

Returns
The position of the closest potential start codon, or None if not found.
Return type
Optional[int]

`_count_lof_vars`(*seqvar: SeqVar*, *start_pos: int*, *end_pos: int*) → Tuple[int, int]

Counts Loss-of-Function (LoF) variants in the specified range.

The method retrieves variants from the specified range and iterates through the available data of each variant. The method counts the number of LoF variants and the number of frequent LoF variants in the specified range, based on the gnomAD genomes data (for the consequence of Nonsense and Frameshift variants) and the allele frequency (for the frequency of the LoF variants in the general population).

Note: A LoF variant is considered frequent if its occurrence in the general population exceeds some threshold. We use a threshold of 0.1% to determine if the LoF variant is frequent.

Parameters
- `seqvar` – The sequence variant being analyzed.
- `start_pos` – The start position of the range.
- `end_pos` – The end position of the range.

Returns
The number of frequent LoF variants and the total number of LoF variants.
Return type
Tuple[int, int]

`_count_pathogenic_vars`(*seqvar: SeqVar*, *start_pos: int*, *end_pos: int*) → Tuple[int, int]

Counts pathogenic variants in the specified range.

The method retrieves variants from the specified range and iterates through the ClinVar data of each variant to count the number of pathogenic variants and the total number of variants.

Parameters
- `seqvar` – The sequence variant being analyzed.
- `start_pos` – The start position of the range.
- `end_pos` – The end position of the range.

Returns
The number of pathogenic variants and the total number of variants.
Return type
Tuple[int, int]
Raises
`InvalidAPIResposeError` – If the API response is invalid or cannot be processed.

`_find_aff_exon_pos`(*var_pos: int*, *exons: List[Exon]*) → Tuple[int, int]

Find start and end positions of the affected exon.

Parameters
- `var_pos` – The position of the variant in the coding sequence.
- `exons` – A list of exons of the gene where the variant occurs.

Returns
The start and end positions of the affected exon.
Return type
Tuple[int, int]
Raises
`AlgorithmError` – If the affected exon is not found.

`_get_conseq`(*val: SeqVarPVS1Consequence*) → List[str]

Get the VEP consequence of the sequence variant by value.

Parameters
val – The value of the consequence.
Returns
The VEP consequences of the sequence variant.
Return type
List[str]

_skipping_exon_pos(*seqvar: SeqVar*, *exons: List[Exon]*) → Tuple[int, int]

Calculate the length of the closest to the seqvar exon.

The method calculates the length of the exon, which can be skipped due to the variant consequences.
Parameters
- **seqvar** – The sequence variant being analyzed.
- **exons** – A list of exons of the gene where the variant occurs.

Returns
The start and end positions of the exon skipping region.
Return type
Tuple[int, int]

alt_start_cdn(*cds_info: Dict[str, CdsInfo]*, *hgvs: str*) → bool

Check if the variant introduces an alternative start codon in other transcripts.
Implementation of the rule:
- **Iterating through all transcripts and checking if the coding sequence start** differs from the main transcript.
- If the start codon differs, the rule is met.

Note:
Rule:
If the variant introduces an alternative start codon in other transcripts, it is considered to be non-pathogenic.

Parameters
- **hgvs** – The main transcript ID.
- **cds_info** – A dictionary containing the CDS information for all transcripts.

Returns
True if the variant introduces an alternative start codon in other transcripts, False otherwise.
Return type
bool

annonars_client

Annonars client for the API.

comment_pvs1: str

Comment to store the prediction explanation.

crit4prot_func(*seqvar: SeqVar*, *exons: List[Exon]*, *strand: GenomicStrand*) → bool

Checks if the truncated or altered region is critical for the protein function.

This method assesses the impact of a sequence variant based on the presence of pathogenic variants downstream of the new stop codon, utilizing both experimental and clinical evidence.

Implementation of the rule: - Calculating the range of the altered region, based on the position of the variant in the coding sequence and the exons of the gene. - Fetching variants from the specified range of the altered region. - Counting the number of pathogenic variants in that region, by iterating through

the clinvar data of each variant. - Considering the region critical if the frequency of pathogenic variants exceeds 5%.

Note: The significance of a truncated or altered region is determined by the presence and frequency of pathogenic variants downstream from the new stop codon. We use a threshold of 5% to determine if the region is critical for the protein function.

Parameters

- **seqvar** – The sequence variant being analyzed.
- **cds_pos** – The position of the variant in the coding sequence.
- **exons** – A list of exons of the gene where the variant occurs.
- **strand** – The genomic strand of the gene.

Returns

True if the altered region is critical for the protein function, otherwise False.

Return type

bool

Raises

InvalidAPIResponseError – If the API response is invalid or cannot be processed.

exon_skip_or_cryptic_ss_disrupt(*seqvar: SeqVar*, *exons: List[Exon]*, *consequences: List[str]*, *strand: GenomicStrand*) → bool

Check if the variant causes exon skipping or cryptic splice site disruption.

The method checks if the variant causes exon skipping or cryptic splice site disruption based on the position of the variant in the coding sequence and the exons of the gene.

Implementation of the rule: - If the exon length is not a multiple of 3, the variant is predicted to cause exon skipping. - If the variant is a splice acceptor or donor variant, the method predicts cryptic splice site disruption.

Note: Rule: If the variant causes exon skipping or cryptic splice site disruption, it is considered to be pathogenic.

Parameters

- **seqvar** – The sequence variant being analyzed.
- **exons** – A list of exons of the gene where the variant occurs.
- **consequences** – A list of VEP consequences of the sequence variant.

Returns

True if the variant causes exon skipping or cryptic splice site disruption,
False if preserves reading frame.

Return type

bool

in_bio_relevant_tx(*transcript_tags: List[str]*) → bool

Checks if the exon with SeqVar is in a biologically relevant transcript.

Implementation of the rule:

If the variant is located in a transcript with a MANE Select tag, it is considered to be in a biologically relevant transcript.

Parameters

transcript_tags – A list of tags for the transcript.

Returns

True if the variant is in a biologically relevant transcript, False otherwise.

Return type

bool

lof_freq_in_pop(*seqvar: SeqVar*, *exons: List[Exon]*, *strand: GenomicStrand*) → bool

Checks if the Loss-of-Function (LoF) variants in the exon are frequent in the general population.

This function determines the frequency of LoF variants within a specified genomic region and evaluates whether this frequency exceeds a defined threshold indicative of common occurrence in the general population.

Implementation of the rule: - Calculating the range of the altered region (coding sequence of the transcript). - Counting the number of LoF variants and frequent LoF variants in that region. - Considering the LoF variants frequent in the general population if the frequency of "frequent" LoF variants exceeds 10%.

Note: A LoF variant is considered frequent if its occurrence in the general population exceeds some threshold. We use a threshold of 10% to determine if the LoF variant is frequent.

Parameters

- **seqvar** – The sequence variant being analyzed.
- **exons** – A list of exons of the gene where the variant occurs.
- **strand** – The genomic strand of the gene.

Returns

True if the LoF variant frequency is greater than 10%, False otherwise.

Return type

bool

Raises

InvalidAPIResponseError – If the API response is invalid or cannot be processed.

lof_rm_gt_10pct_of_prot(*prot_pos: int*, *prot_length: int*) → bool

Check if the LoF variant removes more than 10% of the protein.

The method checks if the LoF variant removes more than 10% of the protein based on the position of the variant in the protein and the length of the protein.

Note:

Rule:

A LoF variant is considered to remove more than 10% of the protein if the variant removes more than 10% of the protein.

Parameters

- **prot_pos** – The position of the variant in the protein.
- **prot_length** – The length of the protein.

Returns

True if the LoF variant removes more than 10% of the protein, False otherwise.

Return type

bool

undergo_nmd(*var_pos: int*, *hgnc_id: str*, *strand: GenomicStrand*, *exons: List[Exon]*) → bool

Classifies if the variant undergoes Nonsense-mediated decay (NMD).

Implementation of the rule: The method checks if the variant is in the GJB2 gene and always predicts it to undergo NMD. If the variant is not in the GJB2 gene, the method checks if the new stop codon position including the 5' UTR length is less or equal to the NMD cutoff, and if it is indeed less or equal, the variant is predicted to undergo NMD. Otherwise, the variant is predicted to escape NMD.

Note:

Rule:

If the variant is located in the last exon or in the last 50 nucleotides of the penultimate exon, it is NOT predicted to undergo NMD.

Important:
For the GJB2 gene (HGNC:4284), the variant is always predicted to undergo NMD.

Parameters
- **var_pos** – The position of the new stop codon !including the 5' UTR length!.
- **hgnc_id** – The HGNC ID of the gene.
- **strand** – The genomic strand of the gene.
- **exons** – A list of exons of the gene.

Returns
True if the variant undergoes NMD, False if variant escapes NMD.

Return type
bool

up_pathogenic_vars(*seqvar: SeqVar*, *exons: List[Exon]*, *strand: GenomicStrand*) → bool

Look for pathogenic variants upstream of the closest potential in-frame start codon.

The method checks for pathogenic variants upstream of the closest potential in-frame start codon. The method is implemented as follows: - Find the closest potential in-frame start codon. - Fetch and count pathogenic variants in the specified range. - Return True if pathogenic variants are found, otherwise False.

Parameters
- **seqvar** – The sequence variant being analyzed.
- **cds_pos** – The position of the variant in the coding sequence.
- **exons** – A list of exons of the gene where the variant occurs.
- **strand** – The genomic strand of the gene.

Returns
True if pathogenic variants are found upstream of the closest potential in-frame start codon, False otherwise.

Return type
bool

2.4.2 PS1 and PM5

Implementation of PS1 and PM5 prediction for sequence variants.

class src.seqvar.auto_ps1_pm5.**AutoPS1PM5**

Bases: AutoACMGHelper

Class for PS1 and PM5 prediction.

_affect_splicing(*var_data: AutoACMGSeqVarData*) → bool

Check if the variant affects splicing. If any of spliceAI scores are above the threshold, the variant is considered to affect splicing.

Parameters
var_data – The variant information.

Returns
True if the variant affects splicing, False otherwise.

Return type
bool

_get_var_info(*seqvar: SeqVar*) → AnnonarsVariantResponse | None

Get variant information from Annonars.

Parameters
seqvar – The sequence variant.

Returns
Annonars response if the variant is found, None otherwise.

Return type
Optional[AnnonarsVariantResponse]

_is_missense(*var_data: AutoACMGSeqVarData*) → bool

Check if the variant's consequence is missense.

Parameters
var_data – The variant information.
Returns
True if the variant is a missense variant, False otherwise.
Return type
bool

_is_pathogenic(*variant_info: VariantResult*) → bool

Check if the variant is pathogenic based on ClinVar data.

Parameters
variant_info – Annonars variant information
Returns
True if the variant is pathogenic
Return type
bool

_is_splice_affecting(*var_data: AutoACMGSeqVarData*) → bool

Check if the variant is a splice-affecting variant.

Parameters
var_data – The variant information.
Returns
True if the variant is a splice-affecting variant, False otherwise.
Return type
bool

_parse_HGVSp(*pHGVSp: str*) → AminoAcid | None

Parse the pHGVSp from VEP into its components.

Parameters
pHGVSp – The protein change in HGVS format.
Returns
The amino acid change if the pHGVSp is valid, None otherwise.
Return type
Optional[AminoAcid]

annonars_client

Annonars client for the API.

comment_ps1pm5: str

Comment to store the prediction explanation.

predict_ps1pm5(*seqvar: SeqVar*, *var_data: AutoACMGSeqVarData*) → Tuple[AutoACMGCriteria, AutoACMGCriteria]

Predict PS1 and PM5 criteria.

prediction_ps1pm5: PS1PM5 | None

Prediction result.

verify_ps1pm5(*seqvar: SeqVar*, *var_data: AutoACMGSeqVarData*) → Tuple[PS1PM5 | None, str]

Predicts the criteria PS1 and PM5 for the provided sequence variant.

Implementation of the rule PS1 and PM5:

The method implements the rule by: - Getting the primary variant information & parsing the primary

amino acid change. - Iterating over all possible alternative bases & getting the alternative variant information. - Parsing the alternative amino acid change & checking if the alternative variant is pathogenic. - If the alternative variant is pathogenic and the amino acid change is the same as the primary variant, then PS1 is set to True. - If the alternative variant is pathogenic and the amino acid change is different from the primary variant, then PM5 is set to True.

Note: Rules: PS1: Same amino acid change as a previously established pathogenic variant regardless of nucleotide change. PM5: Novel missense change at an amino acid residue where a different missense change determined to be pathogenic has been seen before.

Returns
Prediction result and the comment with the explanation.
Return type
Tuple[Optional[PS1PM5], str]

src.seqvar.auto_ps1_pm5.**DNA_BASES = ['A', 'C', 'G', 'T']**

DNA bases

src.seqvar.auto_ps1_pm5.**REGEX_HGVSP = re.compile('p\\.(\\D+)(\\d+)(\\D+)')**

Regular expression for parsing pHGVSp

2.4.3 PM1

Implementation of PM1 criteria.

class src.seqvar.auto_pm1.**AutoPM1**

Bases: AutoACMGHelper

Class for PM1 prediction.

_count_vars(*seqvar: SeqVar*, *start_pos: int*, *end_pos: int*) → Tuple[int, int]

Counts pathogenic and benign variants in the specified range.

The method retrieves variants from the specified range and iterates through the ClinVar data of each variant to count the number of pathogenic and benign SNVs.

Note: The method considers "Pathogenic" and "Likely pathogenic" as pathogenic and "Benign" and "Likely benign" as benign.

Parameters
- **seqvar** – The sequence variant being analyzed.
- **start_pos** – The start position of the range.
- **end_pos** – The end position of the range.

Returns
The number of pathogenic and benign variants.
Return type
Tuple[int, int]
Raises
- **AlgorithmError** – If end position is less than the start position.
- **InvalidAPIResposeError** – If the API response is invalid or cannot be processed.

_get_affected_exon(*var_data: AutoACMGSeqVarData*, *seqvar: SeqVar*) → int

Get the affected exon number for the variant.

Go through all exons and count them before the variant position. The method also considers the strand of the gene.

Parameters

- **var_data** – AutoACMGData object
- **seqvar** – SeqVar object

Returns

Affected exon number

Return type

int

Raises

AlgorithmError – If the strand is invalid.

_get_uniprot_domain(*seqvar: SeqVar*) → Tuple[int, int] | None

Retrieve the UniProt domain for the variant and return the start and end positions if found or None otherwise.

Parameters

seqvar – The sequence variant being analyzed.

Returns

The start and end positions of the UniProt domain if found, None otherwise.

Return type

Optional[Tuple[int, int]]

Raises

AlgorithmError – If tabix fails to query the UniProt file.

annonars_client

Annonars client for the API.

comment_pm1: str

comment_pm1 to store the prediction explanation.

predict_pm1(*seqvar: SeqVar*, *var_data: AutoACMGSeqVarData*) → AutoACMGCriteria

Predict PM1 criteria.

prediction_pm1: PM1 | None

Prediction result.

verify_pm1(*seqvar: SeqVar*, *var_data: AutoACMGSeqVarData*) → Tuple[PM1 | None, str]

Predict PM1 criteria.

The method verifies the PM1 criteria for the variant. It first checks if the variant is in the mitochondrial genome. If so, it returns PM1 as not met. Otherwise, it counts the number of pathogenic and benign variants in the range of 25 bases before and after the variant position. If the number of pathogenic variants is greater than or equal to the threshold, it returns PM1 as met. Otherwise, it retrieves the UniProt domain for the variant and counts the number of pathogenic and benign variants in the domain. If the number of pathogenic variants is greater than or equal to 1/4 of the domain length, it returns PM1 as met. Otherwise, it returns PM1 as not met.

Parameters

- **seqvar** – The sequence variant being analyzed.
- **var_data** – AutoACMGData object

Returns

The prediction result and the explanation

Return type

Tuple[Optional[PM1], str]

2.4.4 PM2, BA1, BS1, BS2

Implementation of BA1, BS1, BS2, PM2 prediction for sequence variants.

class src.seqvar.auto_pm2_ba1_bs1_bs2.**AutoPM2BA1BS1BS2**

Bases: AutoACMGHelper

Class for PM2, BA1, BS1, BS2 prediction.

_ba1_exception(*seqvar: SeqVar*) → bool

Check the exception for BA1 criteria, specified by VCEP modification. If the variant in the exception list, return True.

Parameters
seqvar – The sequence variant.

Returns
True if the variant is in exception list.

Return type
bool

_bs2_not_applicable(*var_data: AutoACMGSeqVarData*) → bool

Check if the BS2 criteria is not applicable.

Per default, the BS2 criteria is applicable. Only some specific VCEP modifications can exclude the BS2 criteria.

Parameters
seqvar – The sequence variant.

Returns
True if the BS2 criteria is not applicable.

Return type
bool

_check_zyg(*seqvar: SeqVar*, *var_data: AutoACMGSeqVarData*) → bool

Check the zygosity of the sequence variant.

If the variant is mitochondrial, it is not considered for BS2 criteria. Otherwise, parse the allele condition and check the zygosity: If the variant is on X chromosome, check the allele count for XX and XY as follows: - for dominant: XX allele count - 2 * XX nhomalt + XY allele count > 2 - for recessive: XX nhomalt + XY nhomalt > 2 - for dominant/recessive: XX allele count - 2 * XX nhomalt + XY allele count > 2 and XX nhomalt + XY nhomalt > 2 If the variant is on autosomal chromosomes, check the allele count as follows: - for dominant: allele count - 2 * nhomalt > 5 - for recessive: nhomalt > 5 - for dominant/recessive: allele count - 2 * nhomalt > 5 and nhomalt > 5 Return True if the variant is in a recessive (homozygous), dominant (heterozygous), or X-linked (hemizygous) disorder (condition is met).

Parameters
variant_data – The variant data.

Returns
True if the variant is recessive (homozygous), dominant (heterozygous), or X-linked (hemizygous) disorder.

Return type
bool

_get_af(*seqvar: SeqVar*, *var_data: AutoACMGSeqVarData*) → float | None

Get the allele frequency for the sequence variant.

Parameters
- **seqvar** – The sequence variant.
- **variant_data** – The variant data.

Returns
The allele frequency. None if no controls data

Return type
Optional[float]

_get_allele_cond(*seqvar: SeqVar*) → AlleleCondition

Get the allele condition for the sequence variant.

Get the Clingen dosage for the gene from the gene transcript data (mehari). If the Clingen dosage is unknown, try Decipher and Domino scores. Compare the scores to specific thresholds to determine the allele condition.

Parameters
seqvar – The sequence variant.
Returns
The allele condition.
Return type
AlleleCOndition

_get_any_af(*var_data: AutoACMGSeqVarData*) → AlleleCount | None

Get the highest allele frequency information for any population. The control group has priority.

Parameters
var_data – The variant data.
Returns
The highest allele frequency for any population. None if no data found.
Return type
Optional[AlleleCount]

_get_control_af(*var_data: AutoACMGSeqVarData*) → AlleleCount | None

Get the allele frequency information for the control population.

Parameters
var_data – The variant data.
Returns
The allele frequency for the control population. None if no data found.
Return type
Optional[AlleleCount]

_get_m_af(*var_data: AutoACMGSeqVarData*) → float | None

Get the allele frequency for the mitochondrial sequence variant.

Parameters
variant_data – The variant data.
Returns
The allele frequency. None if no controls data
Return type
Optional[float]

annonars_client

Annonars client for the API.

comment_pm2ba1bs1bs2: str

comment_pm2ba1bs1bs2 to store the prediction explanation.

predict_pm2ba1bs1bs2(*seqvar: SeqVar*, *var_data: AutoACMGSeqVarData*) → Tuple[AutoACMGCriteria, AutoACMGCriteria, AutoACMGCriteria, AutoACMGCriteria]

Predict PM2, BA1, BS1, BS2 criteria.

prediction_pm2ba1bs1bs2: PM2BA1BS1BS2 | None

Prediction result.

verify_pm2ba1bs1bs2(*seqvar: SeqVar*, *var_data: AutoACMGSeqVarData*) → Tuple[PM2BA1BS1BS2 | None, str]

Predicts the PM2, BA1, BS1, BS2 criteria for the sequence variant.

Predict criteria by checking the allele frequency data and comparing it to the thresholds. Assign PM2, BA1 and BS1 criteria based on the allele frequency data. For BS2 criteria, check zygosity of the variant.

Note: Rules: PM2: Absent from controls allele frequency data.

BA1: Allele frequency is >5%.

BS1: Allele frequency is between 1% and 5%.

BS2: Observed in a healthy adult individual for a recessive (homozygous), dominant (heterozygous), or X-linked (hemizygous) disorder, with full penetrance expected at an early age.

Parameters
- **seqvar** – The sequence variant.
- **var_data** – The variant data.

Returns
The prediction result and the explanation.

Return type
Tuple[Optional[PM2BA1BS1BS2], str]

2.4.5 PM4 and BP3

Implementation of PM4 and BP3 rules for sequence variants.

class src.seqvar.auto_pm4_bp3.**AutoPM4BP3**

Bases: AutoACMGHelper

Class for PM4 and BP3 prediction.

_bp3_not_applicable(*seqvar: SeqVar*, *var_data: AutoACMGSeqVarData*) → bool

Check if BP3 is not applicable for the variant.

Parameters
- **seqvar** – Sequence variant.
- **var_data** – The variant information.

Returns
True if BP3 is not applicable, False otherwise.

Return type
bool

_in_repeat_region(*seqvar: SeqVar*) → bool

Check if the variant is in a repeat region using the RepeatMasker track.

Parameters
seqvar – Sequence variant.

Returns
True if the variant is in a repeat region, False otherwise.

Return type
bool

Raises
AlgorithmError – If tabix fails to query the RepeatMasker track.

_is_stop_loss(*var_data: AutoACMGSeqVarData*) → bool

Check if the variant's consequence is a stop-loss.

Parameters
var_data – The variant information.

Returns
True if the variant is a stop-loss variant, False otherwise.

Return type
bool

annonars_client

Annonars client for the API.

comment_pm4bp3: str

Comment to store the prediction explanation.

is_inframe_delins(*var_data: AutoACMGSeqVarData*) → bool

Check if the variant's consequence is an in-frame deletion/insertion.

Parameters
var_data – The variant information.

Returns
True if the variant is an in-frame deletion/insertion, False otherwise.

Return type
bool

predict_pm4bp3(*seqvar: SeqVar*, *var_data: AutoACMGSeqVarData*) → Tuple[AutoACMGCriteria, AutoACMGCriteria]

Predict PM4 and BP3 criteria.

prediction_pm4bp3: PM4BP3 | None

Prediction result.

verify_pm4bp3(*seqvar: SeqVar*, *var_data: AutoACMGSeqVarData*) → Tuple[PM4BP3 | None, str]

Predicts PM4 and BP3 criteria for the provided sequence variant.

Implementation of the rule: - If the variant is a stop-loss variant, PM4 is True and BP3 is False. - If the variant is an in-frame deletion/insertion: - If the variant is not in a repeat region, PM4 is True and BP3 is False. - If the variant is in a repeat region, PM4 is False and BP3 is True. - Otherwise, PM4 and BP3 are False.

Note: Rules: PM4: Protein length changes due to in-frame deletions/insertions in a non-repeat region or stop-loss variants. BP3: In-frame deletions/insertions in a repetitive region without a known function.

Parameters
- **seqvar** – Sequence variant.
- **var_data** – The variant information

Returns
Prediction result and comment.

Return type
Tuple[Optional[PM4BP3], str]

2.4.6 PP2 and BP1

Implementation of PP2 and BP1 criteria.

class src.seqvar.auto_pp2_bp1.**AutoPP2BP1**

Bases: AutoACMGHelper

Class for PP2 and BP1 prediction.

_get_missense_vars(*seqvar: SeqVar*, *start_pos: int*, *end_pos: int*) → Tuple[int, int, int]

Counts pathogenic, benign, and total missense variants in the specified range.

The method retrieves variants from the specified range and iterates through the ClinVar data of each variant to count the number of pathogenic variants, benign variants, and the total number of missense variants.

Parameters

- **seqvar** – The sequence variant being analyzed.
- **start_pos** – The start position of the range.
- **end_pos** – The end position of the range.

Returns

The number of pathogenic variants, benign variants, and the total number of missense variants.

Return type

Tuple[int, int, int]

Raises

- **AlgorithmError** – If end position is less than the start position.
- **InvalidAPIResposeError** – If the API response is invalid or cannot be processed.

_is_missense(*var_data: AutoACMGSeqVarData*) → bool

Check if the variant is a missense variant.

Parameters

var_data – The variant information.

Returns

True if the variant is a missense variant, False otherwise.

Return type

bool

annonars_client

Annonars client for the API.

comment_pp2bp1: str

Comment to store the prediction explanation.

predict_pp2bp1(*seqvar: SeqVar*, *var_data: AutoACMGSeqVarData*) → Tuple[AutoACMGCriteria, AutoACMGCriteria]

Predict PP2 and BP1 criteria,

prediction_pp2bp1: PP2BP1 | None

Prediction result.

verify_pp2bp1(*seqvar: SeqVar*, *var_data: AutoACMGSeqVarData*) → Tuple[PP2BP1 | None, str]

Predict PP2 and BP1 criteria.

The method verifies the PP2 and BP1 criteria for the provided sequence variant. It checks if the variant is a missense variant and assigns PP2 and BP1 based on the missense Z-score. If the Z-score is not available, the method counts the pathogenic and benign missense variants in the range of the variant and assigns PP2 and BP1 based on the ratio of pathogenic and benign variants.

Parameters
- **seqvar** – The sequence variant being analyzed.
- **var_data** – The variant information.

Returns
The prediction result and the explanation.

Return type
Tuple[Optional[PP2BP1], str]

2.4.7 PP3 and BP4

Implementation of the PP3 and BP4 criteria.

class src.seqvar.auto_pp3_bp4.**AutoPP3BP4**

Bases: AutoACMGHelper

Class for PP3 and BP4 prediction.

_affect_spliceAI(*var_data: AutoACMGSeqVarData*) → bool

Predict splice site alterations using SpliceAI.

If any of SpliceAI scores are greater than specific thresholds, the variant is considered to affect splicing.

Parameters
var_data – The data containing variant scores and thresholds.

Returns
True if the variant is a splice site alteration, False otherwise.

Return type
bool

_is_benign_score(*var_data: AutoACMGSeqVarData*, **score_threshold_pairs: Tuple[str, float]*) → bool

Check if any of the specified scores meet their corresponding threshold.

Parameters
- **var_data** – Variant data containing scores and thresholds.
- **score_threshold_pairs** – Pairs of score attributes and their corresponding benign
- **thresholds.**

Returns
True if any of the specified scores meet their corresponding threshold, False otherwise.

Return type
bool

_is_benign_splicing(*var_data: AutoACMGSeqVarData*) → bool

Check if the variant is benign based on splicing scores.

Checks if the Ada and RF scores are less than the thresholds.

Parameters
var_data – The variant information.

Returns
True if the variant is benign, False otherwise.

Return type
bool

Raises
MissingDataError – If the Ada and RF scores are missing.

_is_inframe_indel(*var_data: AutoACMGSeqVarData*) → bool

Check if the variant's consequence is an inframe indel.

Parameters
var_data – The variant information.
Returns
True if the variant is an inframe indel, False otherwise.
Return type
bool

_is_intron_variant(*var_data: AutoACMGSeqVarData*) → bool

Check if the variant's consequence is an intron variant.
Parameters
var_data – The variant information.
Returns
True if the variant is an intron variant, False otherwise.
Return type
bool

_is_missense_variant(*var_data: AutoACMGSeqVarData*) → bool

Check if the variant's consequence is a missense variant.
Parameters
var_data – The variant information.
Returns
True if the variant is a missense variant, False otherwise.
Return type
bool

_is_pathogenic_score(*var_data: AutoACMGSeqVarData*, **score_threshold_pairs: Tuple[str, float]*) → bool

Check if any of the specified scores meet their corresponding threshold.
Parameters
- **var_data** – Variant data containing scores and thresholds.
- **score_threshold_pairs** – Pairs of score attributes and their corresponding pathogenic
- **thresholds.**

Returns
True if any of the specified scores meet their corresponding threshold, False otherwise.
Return type
bool

_is_pathogenic_splicing(*var_data: AutoACMGSeqVarData*) → bool

Check if the variant is pathogenic based on splicing scores.

Checks if the Ada and RF scores are greater than the thresholds.
Parameters
var_data – The variant information.
Returns
True if the variant is pathogenic, False otherwise.
Return type
bool
Raises
MissingDataError – If the Ada and RF scores are missing.

_is_splice_variant(*var_data: AutoACMGSeqVarData*) → bool

Check if the variant's consequence is a splice related.
Parameters
var_data – The variant information.

Returns
True if the variant is a splice variant, False otherwise.
Return type
bool

_is_synonymous_variant(*var_data: AutoACMGSeqVarData*) → bool

Check if the variant's consequence is a synonymous variant.
Parameters
var_data – The variant information.
Returns
True if the variant is a synonymous variant, False otherwise.
Return type
bool

_is_utr_variant(*var_data: AutoACMGSeqVarData*) → bool

Check if the variant's consequence is an UTR variant.
Parameters
var_data – The variant information.
Returns
True if the variant is an UTR variant, False otherwise.
Return type
bool

annonars_client

Annonars client for the API.

comment_pp3bp4: str

Comment to store the prediction explanation.

predict_pp3bp4(*seqvar: SeqVar*, *var_data: AutoACMGSeqVarData*) → Tuple[AutoACMGCriteria, AutoACMGCriteria]

Predict PP3 and BP4 criteria.

prediction_pp3bp4: PP3BP4 | None

Prediction result.

verify_pp3bp4(*seqvar: SeqVar*, *var_data: AutoACMGSeqVarData*) → Tuple[PP3BP4 | None, str]

Predict PP3 and BP4 criteria.

The method checks the variant's pathogenicity based on the provided scores and thresholds. First of all it checks the pathogenic and benign scores against the thresholds if the default strategy is used. Otherwise, it checks the pathogenic and benign scores against the specified thresholds and then checks the splicing scores. If the variant is a splice site alteration or has the pathogenic score, the variant is considered pathogenic. If the variant doesn't affect splicing and has the benign score, the variant is considered benign.

Note: The non-default assesment strategy is used for some VCEPs.

Parameters
- **seqvar** – Sequence variant.
- **var_data** – The variant information.

Returns
The prediction result and the comment.
Return type
Tuple[Optional[PP3BP4], str]

2.4.8 BP7

Implementation of BP7 criteria.

class src.seqvar.auto_bp7.**AutoBP7**

Bases: AutoACMGHelper

Class for BP7 prediction.

_affect_canonical_ss(*seqvar: SeqVar*, *var_data: AutoACMGSeqVarData*) → bool

Predict if the variant affects canonical splice site.

Check if the variant position is within +1/-2 of the start/end of an intron. Note, that for the minus strand, the donor site is at the start of the intron and the acceptor site is at the end of the intron.

Parameters
- **seqvar** – The sequence variant.
- **var_data** – The variant data.

Returns
True if the variant affects canonical splice site, False otherwise.

Return type
bool

Raises
MissingDataError – If the strand information is missing.

_is_bp7_exception(*seqvar: SeqVar*, *var_data: AutoACMGSeqVarData*) → bool

Help function to check if the variant is an exception.

Per default there are no exceptions for BP7. This function can be overridden in the special VCEP implementations.

Parameters
- **seqvar** – The sequence variant.
- **var_data** – The variant data.

Returns
True if the variant is an exception, False otherwise.

Return type
bool

_is_conserved(*var_data: AutoACMGSeqVarData*) → bool

Predict if the variant is conserved.

Check if the variant is conserved using the phyloP100 score.

Parameters
variant_info – The variant information.

Returns
True if the variant is not conserved, False otherwise.

Return type
bool

_is_intronic(*var_data: AutoACMGSeqVarData*) → bool

Predict if the variant is intronic.

Check if the variant's consequence is intronic.

Parameters
variant_info – The variant information.

Returns
True if the variant is intronic, False otherwise.

Return type
bool

_is_synonymous(*var_data: AutoACMGSeqVarData*) → bool

Predict if the variant is synonymous.

Check if the variant's consequence is synonymous.

Parameters
variant_info – The variant information.

Returns
True if the variant is synonymous, False otherwise.

Return type
bool

_spliceai_impact(*var_data: AutoACMGSeqVarData*) → bool

Predict splice site alterations using SpliceAI.

If any of SpliceAI scores are greater than specific thresholds, the variant is predicted to affect splicing.

Parameters
var_data – The data containing variant scores and thresholds.

Returns
True if the variant is a splice site alteration, False otherwise.

Return type
bool

annonars_client

Annonars client for the API.

comment_bp7: str

Comment to store the prediction explanation.

predict_bp7(*seqvar: SeqVar*, *var_data: AutoACMGSeqVarData*) → AutoACMGCriteria

Predict BP7 criterion.

prediction_bp7: BP7 | None

Prediction result.

verify_bp7(*seqvar: SeqVar*, *var_data: AutoACMGSeqVarData*) → Tuple[BP7 | None, str]

Predict BP7 criterion.

Predict if the variant meets the BP7 criterion by: - Checking if the variant is in the mitochondrial genome. If so, BP7 is not met. - Checking if the variant is synonymous and not conserved and not predicted to affect splicing or intronic variant that does not affect canonical splice site and also not conserved and not predicted to affect splicing. If so, BP7 is met.

Note: Some VCEP implementations might have exceptions for BP7. In that case, the prediction avoids the default checks and returns that BP7 is not met.

Parameters
- **seqvar** – The sequence variant.
- **var_data** – The variant data.

Returns
The prediction and the comment.

Return type
Tuple[Optional[BP7], str]

2.5 Implementation of different ACMG criteria for structural variants

This section contains the internal building blocks for the implementation of the different ACMG criteria for structural variants.

2.5.1 PVS1

For structural variants: PVS1 criteria for Structural Variants (StrucVar).

class src.strucvar.auto_pvs1.**AutoPVS1**

Bases: *StrucVarHelper*

Handles the PVS1 criteria assesment for structural variants.

_calc_cds(*exons: List[Exon]*, *strand: GenomicStrand*, *start_codon: int*, *stop_codon: int*) → List[Exon]

Remove UTRs from exons.

Parameters

- **exons** – List of exons for the gene.
- **strand** – The genomic strand of the gene.
- **start_codon** – Position of the start codon.
- **stop_codon** – Position of the stop codon.

Returns

List of exons without UTRs.

Return type

List[Exon]

Raises

MissingDataError – If the genomic strand is not set.

_count_lof_vars(*strucvar: StrucVar*) → Tuple[int, int]

Counts Loss-of-Function (LoF) variants within the range of a structural variant.

The method retrieves variants from the range defined by the structural variant's start and stop positions and iterates through the available data of each variant to count the number of LoF variants and the number of frequent LoF variants, based on the gnomAD genomes data (for the consequences of Nonsense and Frameshift variants) and the allele frequency (for the frequency of the LoF variants in the general population).

Note: A LoF variant is considered frequent if its occurrence in the general population exceeds a threshold of 0.1%.

Parameters

strucvar – The structural variant being analyzed.

Returns

The number of frequent LoF variants and the total number of LoF variants.

Return type

Tuple[int, int]

Raises

- **AlgorithmError** – If the end position is less than the start position.
- **InvalidAPIResposeError** – If the API response is invalid or cannot be processed.

_count_pathogenic_vars(*strucvar: StrucVar*) → Tuple[int, int]

Counts pathogenic variants in the range specified by the structural variant.

The method retrieves variants from the range defined by the structural variant's start and stop positions and iterates through the ClinVar data of each variant to count the number of pathogenic variants and the total number of variants. The method considers a variant pathogenic if its classification is "Pathogenic" or "Likely pathogenic".

Parameters
strucvar – The structural variant being analyzed.

Returns
The number of pathogenic variants and the total number of variants.

Return type
Tuple[int, int]

Raises
InvalidAPIResposeError – If the API response is invalid or cannot be processed.

_minimal_deletion(*strucvar: StrucVar*, *exons: List[Exon]*) → bool

Check if the variant is a minimal deletion. A minimal deletion affects at least one full exon.

Parameters
- **strucvar** – The structural variant.
- **exons** – The exons of the gene.

Returns
True if the deletion affects at least one full exon, False otherwise.

Return type
bool

Raises
- **AlgorithmError** – If the variant is not a deletion.
- **MissingDataError** – If exons are not available.

annonars_client

Annonars client for the API.

comment_pvs1: str

Comment to store the prediction explanation.

crit4prot_func(*strucvar: StrucVar*) → bool

Check if the deletion is critical for protein function.

This method is implemented by fetching variants from the start to the end of the structural variant, then counting the number of pathogenic variants in that region, by iterating through the clinvar data of each variant. Consider the region critical if the frequency of pathogenic variants exceeds 5%.

Parameters
strucvar – The structural variant being analyzed.

Returns
True if the deletion is critical for protein function, False otherwise.

Return type
bool

Raises
AlgorithmError – If the API response is invalid or cannot be processed.

del_disrupt_rf(*strucvar: StrucVar*, *exons: List[Exon]*, *strand: GenomicStrand*) → bool

Check if the single or multiple exon deletion disrupts the reading frame.

Find the start and end positions of alteration based on the affected exon(s). If the positions lie within the intron(s) of the affected exon(s), the deletion does not disrupt the reading frame. Otherwise, there're two cases: - Check if the deletion starts within an exon. If so, check if the offset from the start of the exon to the start of the deletion is a multiple of 3. If so, the deletion does not disrupt the reading frame. - Check if the deletion stops within an exon. If so, check if the offset from the start of the last affected exon to the stop of the deletion is a multiple of 3. If so, the deletion does not disrupt the reading frame.

Parameters
- **strucvar** – The structural variant.
- **exons** – The exons of the gene.
- **strand** – The genomic strand of the variant.

Returns
True if the deletion disrupts the reading frame, False otherwise.

Raises
- **MissingDataError** – If exons or strand are not available.
- **AlgorithmError** – Less than 1 full exon affected.

static dup_disrupt_rf() → bool

Check if the duplication disrupts the reading frame. NOT IMPLEMENTED!

full_gene_del(*strucvar: StrucVar*, *exons: List[Exon]*) → bool

Check if the variant is a full gene deletion. The deletion affects the whole gene if the start position of the deletion is less than or equal to the start of the first exon and the stop position of the deletion is greater than or equal to the end of the last exon.

Parameters
- **strucvar** – The structural variant.
- **exons** – The exons of the gene.

Returns
True if the variant is a full gene deletion, False otherwise.

Return type
bool

Raises
MissingDataError – If exons are not available.

in_bio_relevant_tsx(*transcript_tags: List[str]*) → bool

Check if the deletion is in a biologically relevant transcript. Check if the transcript has a MANE Select tag.

Parameters
transcript_tags – The tags of the transcript.

Returns
True if the deletion is in a biologically relevant transcript, False otherwise.

Return type
bool

lof_freq_in_pop(*strucvar: StrucVar*) → bool

Checks if the Loss-of-Function (LoF) variants within the structural variant are frequent in the general population.

This function determines the frequency of LoF variants within the range specified by the structural variant and evaluates whether this frequency exceeds a defined threshold indicative of common occurrence in the general population.

Implementation of the rule: - Retrieving the number of LoF variants and frequent LoF variants in the range defined by the structural variant. - Considering the LoF variants frequent in the general population if the frequency of "frequent" LoF variants exceeds 10%.

Note: A LoF variant is considered frequent if its occurrence in the general population exceeds some threshold. We use a threshold of 10% to determine if the LoF variant is frequent.

Parameters
strucvar – The structural variant being analyzed.

Returns
True if the LoF variant frequency is greater than 10%, False otherwise.

Return type
bool

Raises
AlgoritmError – If the API response is invalid or cannot be processed.

lof_rm_gt_10pct_of_prot(*strucvar: StrucVar*, *exons: List[Exon]*, *strand: GenomicStrand*, *start_codon: int*, *stop_codon: int*) → bool

Determine if the deletion removes more than 10% of the protein-coding sequence.

First remove the UTRs from the exons. Then iterate through the CDS exons and calculate the total CDS length and the length of the deleted region. Return True if the deletion removes more than 10% of the protein-coding sequence, False otherwise.

Parameters
- **strucvar** – The structural variant being analyzed.
- **exons** – List of exons for the gene.
- **strand** – The genomic strand of the gene.
- **start_codon** – Position of the start codon.
- **stop_codon** – Position of the stop codon.

Returns
True if the deletion removes more than 10% of the protein, False otherwise.

Return type
bool

Raises
AlgorithmError – If the total CDS length is zero.

predict_pvs1(*strucvar: StrucVar*, *var_data: AutoACMGStrucVarData*) → AutoACMGCriteria

Predict the PVS1 criteria.

static presumed_in_tandem() → bool

Check if the duplication is presumed in tandem. NOT IMPLEMENTED!

static proven_in_tandem() → bool

Check if the duplication is proven in tandem. NOT IMPLEMENTED!

undergo_nmd(*strucvar: StrucVar*, *exons: List[Exon]*, *strand: GenomicStrand*) → bool

Check if the variant undergoes NMD.

Check if the whole deletion affects only the last exon and 50 base pairs of the penultimate exon. If so, the variant does not undergo NMD.

Parameters
- **strucvar** – The structural variant.
- **exons** – The exons of the gene.
- **strand** – The genomic strand of the variant.

Returns
True if the variant undergoes NMD, False otherwise.

Raises
- **MissingDataError** – If exons or strand are not available.
- **AlgorithmError** – If less than 2 exons are available.

verify_pvs1(*strucvar: StrucVar*, *var_data: AutoACMGStrucVarData*) → Tuple[PVS1Prediction, PVS1PredictionStrucVarPath, str]

Make the PVS1 prediction.

The prediction is based on the PVS1 decision tree for structural variants.

Parameters
- **strucvar** – The structural variant.
- **var_data** – The variant information.

Returns
The prediction, prediction path, and the comment.

Return type
Tuple[PVS1Prediction, PVS1PredictionStrucVarPath, str]

class src.strucvar.auto_pvs1.**StrucVarHelper**

Bases: AutoACMGHelper

Helper methods for PVS1 criteria for Structural Variants (StrucVar).

_calc_cds(*exons: List[Exon]*, *strand: GenomicStrand*, *start_codon: int*, *stop_codon: int*) → List[Exon]

Remove UTRs from exons.

Parameters
- **exons** – List of exons for the gene.
- **strand** – The genomic strand of the gene.
- **start_codon** – Position of the start codon.
- **stop_codon** – Position of the stop codon.

Returns
List of exons without UTRs.

Return type
List[Exon]

Raises
MissingDataError – If the genomic strand is not set.

_count_lof_vars(*strucvar: StrucVar*) → Tuple[int, int]

Counts Loss-of-Function (LoF) variants within the range of a structural variant.

The method retrieves variants from the range defined by the structural variant's start and stop positions and iterates through the available data of each variant to count the number of LoF variants and the number of frequent LoF variants, based on the gnomAD genomes data (for the consequences of Nonsense and Frameshift variants) and the allele frequency (for the frequency of the LoF variants in the general population).

Note: A LoF variant is considered frequent if its occurrence in the general population exceeds a threshold of 0.1%.

Parameters
strucvar – The structural variant being analyzed.

Returns
The number of frequent LoF variants and the total number of LoF variants.

Return type
Tuple[int, int]

Raises
- **AlgorithmError** – If the end position is less than the start position.
- **InvalidAPIResposeError** – If the API response is invalid or cannot be processed.

_count_pathogenic_vars(*strucvar: StrucVar*) → Tuple[int, int]

Counts pathogenic variants in the range specified by the structural variant.

The method retrieves variants from the range defined by the structural variant's start and stop positions and iterates through the ClinVar data of each variant to count the number of pathogenic variants and the total number of variants. The method considers a variant pathogenic if its classification is "Pathogenic" or "Likely pathogenic".

Parameters
strucvar – The structural variant being analyzed.

Returns
The number of pathogenic variants and the total number of variants.

Return type
Tuple[int, int]
Raises
InvalidAPIResposeError – If the API response is invalid or cannot be processed.

_minimal_deletion(*strucvar: StrucVar*, *exons: List[Exon]*) → bool

Check if the variant is a minimal deletion. A minimal deletion affects at least one full exon.
Parameters
- **strucvar** – The structural variant.
- **exons** – The exons of the gene.

Returns
True if the deletion affects at least one full exon, False otherwise.
Return type
bool
Raises
- **AlgorithmError** – If the variant is not a deletion.
- **MissingDataError** – If exons are not available.

annonars_client

Annonars client for the API.

comment_pvs1: str

Comment to store the prediction explanation.

crit4prot_func(*strucvar: StrucVar*) → bool

Check if the deletion is critical for protein function.

This method is implemented by fetching variants from the start to the end of the structural variant, then counting the number of pathogenic variants in that region, by iterating through the clinvar data of each variant. Consider the region critical if the frequency of pathogenic variants exceeds 5%.
Parameters
strucvar – The structural variant being analyzed.
Returns
True if the deletion is critical for protein function, False otherwise.
Return type
bool
Raises
AlgorithmError – If the API response is invalid or cannot be processed.

del_disrupt_rf(*strucvar: StrucVar*, *exons: List[Exon]*, *strand: GenomicStrand*) → bool

Check if the single or multiple exon deletion disrupts the reading frame.

Find the start and end positions of alteration based on the affected exon(s). If the positions lie within the intron(s) of the affected exon(s), the deletion does not disrupt the reading frame. Otherwise, there're two cases: - Check if the deletion starts within an exon. If so, check if the offset from the start of the exon to the start of the deletion is a multiple of 3. If so, the deletion does not disrupt the reading frame. - Check if the deletion stops within an exon. If so, check if the offset from the start of the last affected exon to the stop of the deletion is a multiple of 3. If so, the deletion does not disrupt the reading frame.
Parameters
- **strucvar** – The structural variant.
- **exons** – The exons of the gene.
- **strand** – The genomic strand of the variant.

Returns
True if the deletion disrupts the reading frame, False otherwise.
Raises
- **MissingDataError** – If exons or strand are not available.

- **AlgorithmError** – Less than 1 full exon affected.

static dup_disrupt_rf() → bool

Check if the duplication disrupts the reading frame. NOT IMPLEMENTED!

full_gene_del(*strucvar: StrucVar*, *exons: List[Exon]*) → bool

Check if the variant is a full gene deletion. The deletion affects the whole gene if the start position of the deletion is less than or equal to the start of the first exon and the stop position of the deletion is greater than or equal to the end of the last exon.

Parameters
- **strucvar** – The structural variant.
- **exons** – The exons of the gene.

Returns
True if the variant is a full gene deletion, False otherwise.

Return type
bool

Raises
MissingDataError – If exons are not available.

in_bio_relevant_tsx(*transcript_tags: List[str]*) → bool

Check if the deletion is in a biologically relevant transcript. Check if the transcript has a MANE Select tag.

Parameters
transcript_tags – The tags of the transcript.

Returns
True if the deletion is in a biologically relevant transcript, False otherwise.

Return type
bool

lof_freq_in_pop(*strucvar: StrucVar*) → bool

Checks if the Loss-of-Function (LoF) variants within the structural variant are frequent in the general population.

This function determines the frequency of LoF variants within the range specified by the structural variant and evaluates whether this frequency exceeds a defined threshold indicative of common occurrence in the general population.

Implementation of the rule: - Retrieving the number of LoF variants and frequent LoF variants in the range defined by the structural variant. - Considering the LoF variants frequent in the general population if the frequency of "frequent" LoF variants exceeds 10%.

Note: A LoF variant is considered frequent if its occurrence in the general population exceeds some threshold. We use a threshold of 10% to determine if the LoF variant is frequent.

Parameters
strucvar – The structural variant being analyzed.

Returns
True if the LoF variant frequency is greater than 10%, False otherwise.

Return type
bool

Raises
AlgoritmError – If the API response is invalid or cannot be processed.

lof_rm_gt_10pct_of_prot(*strucvar: StrucVar*, *exons: List[Exon]*, *strand: GenomicStrand*, *start_codon: int*, *stop_codon: int*) → bool

Determine if the deletion removes more than 10% of the protein-coding sequence.

First remove the UTRs from the exons. Then iterate through the CDS exons and calculate the total CDS

length and the length of the deleted region. Return True if the deletion removes more than 10% of the protein-coding sequence, False otherwise.

Parameters

- **strucvar** – The structural variant being analyzed.
- **exons** – List of exons for the gene.
- **strand** – The genomic strand of the gene.
- **start_codon** – Position of the start codon.
- **stop_codon** – Position of the stop codon.

Returns

True if the deletion removes more than 10% of the protein, False otherwise.

Return type

bool

Raises

AlgorithmError – If the total CDS length is zero.

static presumed_in_tandem() → bool

Check if the duplication is presumed in tandem. NOT IMPLEMENTED!

static proven_in_tandem() → bool

Check if the duplication is proven in tandem. NOT IMPLEMENTED!

undergo_nmd(*strucvar: StrucVar*, *exons: List[Exon]*, *strand: GenomicStrand*) → bool

Check if the variant undergoes NMD.

Check if the whole deletion affects only the last exon and 50 base pairs of the penultimate exon. If so, the variant does not undergo NMD.

Parameters

- **strucvar** – The structural variant.
- **exons** – The exons of the gene.
- **strand** – The genomic strand of the variant.

Returns

True if the variant undergoes NMD, False otherwise.

Raises

- **MissingDataError** – If exons or strand are not available.
- **AlgorithmError** – If less than 2 exons are available.

2.6 Benchmarking

AutoACMG has undergone rigorous benchmarking against other established tools such as InterVar and GeneBe to ensure accuracy and reliability. This section details the methodology and key findings from our benchmarking exercises.

2.6.1 Methodology

We constructed a specialized dataset derived from the ClinGen Evidence Repository. The dataset and the script used for the benchmarking can be accessed from our GitHub repository:

- **Dataset**: comparison_criteria_custom.csv
- **Benchmarking Script**: Located in the *src/bench* directory of the repository.

The benchmarking process involved:

1. Comparing the predictions made by AutoACMG against those from InterVar and GeneBe using a custom dataset. 2. Utilizing the APIs provided by InterVar and GeneBe to fetch predictions. 3. Computing statistical metrics such as kappa scores, F1 score, precision, and recall to evaluate the performance.

2.6.2 How to Run the Benchmarking Script

To run the benchmarking script, you'll need a development environment set up with the required dependencies. Follow the guide in the Development Environment Setup section to set up the environment. Then run the following command:

```
make bench
```

This command will execute the benchmarking script and generate a *stats.csv* file in the *src/bench* directory. The file contains the statistical metrics computed during the benchmarking process. To analyze the results, you can use the *results_analysis.ipynb* Jupyter notebook provided in the *src/bench* directory. To run the notebook, just execute the following command:

```
make jupyterlab
```

2.6.3 Results

The results of the benchmarking are summarized in the *statistic_metrics.csv* file, which can be viewed here:

- **Statistical Metrics**: statistic_metrics.csv

Key Findings:

- **True Positives: Number of instances where AutoACMG and the comparative tools agreed on a** pathogenic variant.
- **False Positives: Number of instances where AutoACMG predicted a variant as pathogenic, which** was not confirmed by the other tools.
- **False Negatives: Number of instances where AutoACMG did not predict a pathogenic variant, but** the comparative tools did.
- **Kappa Score**: Measures the agreement between AutoACMG and the other tools beyond chance.
- **F1 Score, Precision, and Recall: These metrics provide a more detailed insight into the** accuracy of AutoACMG in identifying pathogenic variants compared to other tools.

2.6.4 Conclusion

The benchmarking results indicate that AutoACMG performs comparably with, if not superior to, other leading tools in the field. The precision and recall rates highlight AutoACMG's ability to reliably identify pathogenic variants, making it a valuable tool for geneticists and researchers. Ongoing improvements and updates will continue to refine its predictions and expand its utility in clinical genomics.

2.7 Usage

This section provides detailed instructions on how to set up and use the Docker container for the AutoACMG system, configure environment variables, and access the API documentation.

2.7.1 Building the Docker Image

To build the Docker image, ensure Docker is installed on your system and navigate to the directory containing the Dockerfile. Use the following command to build the image:

```
docker build -t auto-acmg .
```

This command builds the Docker image with the tag `auto-acmg`. You can replace `auto-acmg` with any other tag suitable for your deployment or versioning schema.

2.7.2 Setting Up Volumes

The AutoACMG Docker container requires access to SeqRepo data directories for sequence information. You must set up volumes that the Docker container can use to access this data without needing to copy it into the container directly. Here's how you set up these volumes:

1. **SeqRepo Data Volume:** This volume stores the SeqRepo datasets.
2. **Custom Project Data Volume:** This volume stores the custom project seqrepo data.

Ensure these directories exist on your host and are populated with the necessary data:

```
mkdir -p /usr/local/share/seqrepo
chown -R root:root /usr/local/share/seqrepo
```

```
pipenv run seqrepo init -i auto-acmg
```

```
pipenv run seqrepo fetch-load -i auto-acmg -n RefSeq NC_000001.10 NC_000002.11 NC_000003.
↪11 \
    NC_000004.11 NC_000005.9 NC_000006.11 NC_000007.13 NC_000008.10 NC_000009.11 NC_
↪000010.10 \
    NC_000011.9 NC_000012.11 NC_000013.10 NC_000014.8 NC_000015.9 NC_000016.9 NC_000017.
↪10 \
    NC_000018.9 NC_000019.9 NC_000020.10 NC_000021.8 NC_000022.10 NC_000023.10 NC_000024.
↪9 \
    NC_012920.1 NC_000001.11 NC_000002.12 NC_000003.12 NC_000004.12 NC_000005.10 NC_
↪000006.12 \
    NC_000007.14 NC_000008.11 NC_000009.12 NC_000010.11 NC_000011.10 NC_000012.12 NC_
↪000013.11 \
    NC_000014.9 NC_000015.10 NC_000016.10 NC_000017.11 NC_000018.10 NC_000019.10 NC_
↪000020.11 \
    NC_000021.9 NC_000022.11 NC_000023.11 NC_000024.10 NC_012920.1
```

Note: The paths used in this example are for demonstration purposes! You should adjust the paths to match your actual directory structure and ensure that the directories have the correct permissions for Docker to access them. The `/usr/local/share/seqrepo` directory is the default location for Linux systems. The `/home/auto-acmg/seqrepo/master` directory is an example of a custom project data directory.

If the above doesn't work for you, you can try to download backups from the CUBI SharePoint. The backups are located in the folder `/Documents/Coding and Engineering/AutoACMG`. Then unarchive them with the following command:

```
tar -czvf seqrepo_local.tar.gz .dev/volumes/seqrepo/local --strip-components=1
tar -czvf seqrepo_master.tar.gz .dev/volumes/seqrepo/master --strip-components=1
```

Finally, you should have the following directories structures in `/usr/local/share/seqrepo` (local): and AUTO_ACMG_SEQREPO_DATA_DIR (master):

```
seqrepo
├── master
│   ├── aliases.sqlite3
│   ├── sequences
│   │       └── db.sqlite3
│   │       └── 2024
│   │             └── 1224
│   │             └── ....
│
└── local
    ├── master
        ├── aliases.sqlite3
        ├── sequences
            └── db.sqlite3
```

2.7.3 Running the Docker Image

Once the Docker image is built, you can run it using the following command to include the volumes:

```
docker run -d -p 8080:8080 --env-file .env \
-v /usr/local/share/seqrepo:/usr/local/share/seqrepo \
-v /home/auto-acmg/seqrepo/master:/home/auto-acmg/seqrepo/master \
auto-acmg
```

This command runs the container in detached mode (in the background), maps port 8080 of the container to port 8080 on the host, making the application accessible via `localhost:8080`. The `--env-file .env` option tells Docker to use the environment variables defined in the `.env` file. Replace `.env` with the path to your actual environment file if different. The `-v` flags map the local directories to their respective directories within the container, ensuring that SeqRepo data is accessible.

Note: You must configure the environment file before running the Docker container and ensure that the directories used for volumes are properly set up and have the correct permissions for Docker to access them. See the configuration details in the sections below.

2.7.4 Configuring the Environment File

The application can be configured using environment variables. An example configuration file named `.env.dev` might look like this:

```
# Disable debug mode per default
DEBUG=0
# Disable cache to avoid memory issues
USE_CACHE=0
```

(continues on next page)

(continued from previous page)

```
# Use the REEV API. Change it if you have other instance of REEV.
API_REEV_URL=https://reev.cubi.bihealth.org/internal/proxy
# Default path to seqrepo data for Docker. Change it to your local development path.
# It can look like this: "/home/<username>/seqrepo/master"
AUTO_ACMG_SEQREPO_DATA_DIR=/home/gromdimon/Custom/seqrepo/master
```

Adjust the values according to your environment. Here are brief descriptions of the variables. Note that not all variables are required for the application to run. More info below.:

- `DEBUG`: Enable or disable debug mode.
- `AUTO_ACMG_USE_CACHE`: Enable or disable caching of API responses.
- `AUTO_ACMG_CACHE_DIR`: Path to the cache directory.
- `API_V1_STR`: Base path for API endpoints.
- `API_REEV_URL`: URL of the REEV API.
- `AUTO_ACMG_API_ANNONARS_URL`: URL of the Annonars API.
- `AUTO_ACMG_API_MEHARI_URL`: URL of the Mehari API.
- `AUTO_ACMG_API_DOTTY_URL`: URL of the Dotty API.
- `AUTO_ACMG_SEQREPO_DATA_DIR`: Path to the project-specific SeqRepo data directory.
- `GENEBE_API_KEY`: API key for the GeneBE service. You'll need it for running the benchmarks.
- `GENEBE_USERNAME`: Username for the GeneBE service. You'll need it for running the benchmarks.

You will most likely need to set the following variables:

- `DEBUG`: Set to 1 to enable debug mode.
- `AUTO_ACMG_USE_CACHE`: Set to 1 to enable caching. This is recommended only for development.
- `AUTO_ACMG_SEQREPO_DATA_DIR`: Set to the path of the custom project SeqRepo data directory.

To pass this configuration to the Docker container, ensure the `.env` file is located where you run the `docker run` command or specify the correct path to the file using the `--env-file` option.

Note: Ensure that the environment variables are correctly set up and that the paths are valid and accessible by the Docker container. The example .env file is provided in .env.dev.

2.7.5 Accessing the OpenAPI Documentation

Once the application is running, you can access the OpenAPI documentation by navigating to:

```
http://localhost:8080/api/v1/docs
```

This URL provides interactive API documentation automatically generated from your OpenAPI specs. It allows you to try out API calls directly from the browser.

2.7.6 API Endpoints

The API provides several endpoints for interacting with the AutoACMG system:

1. **Resolve Variant** Endpoint to resolve a variant based on its name and optionally specify the genome release.
   - **URL**: `/api/v1/resolve`
   - **Method**: `GET`
   - **Parameters**: - `variant_name` (required): The name or identifier of the variant. - `genome_release` (optional): The genome release version, defaults to `GRCh38`.
   - **Success Response**: A JSON object containing resolved variant details.

   Example call:

   ```
   GET /api/v1/resolve?variant_name=chr1:228282272:G:A&genome_release=GRCh38
   ```

2. **Predict Sequence Variant** Endpoint to predict annotations for a sequence variant.
   - **URL**: `/api/v1/predict/seqvar`
   - **Method**: `GET`
   - **Parameters**: - `variant_name` (required): The name or identifier of the sequence variant.
   - **Success Response**: A JSON object containing prediction results.

   Example call:

   ```
   GET /api/v1/predict/seqvar?variant_name=chr1:228282272:G:A
   ```

3. **Predict Structural Variant** Endpoint to predict annotations for a structural variant.
   - **URL**: `/api/v1/predict/strucvar`
   - **Method**: `GET`
   - **Parameters**: - `variant_name` (required): The name or identifier of the structural variant. - `duplication_tandem` (optional): Specifies if the duplication is in tandem.
   - **Success Response**: A JSON object containing structural variant prediction results.

   Example call:

   ```
   GET /api/v1/predict/strucvar?variant_name=chr1:228282272:dup:Tandem
   ```

For more details on the API endpoints and their usage, refer to the OpenAPI documentation accessible at the URL: `http://localhost:8080/api/v1/docs`.

2.8 Dev Quickstart

This document describes how to get started with developing and using the project.

2.8.1 Overview

You will need to have Python 3.12 and pipenv installed. The next step is to checkout the repository and install the Python dependencies. Then, you will be able to utilize the CLI and run the tests. The following assumes a Debian/Ubuntu machine; your mileage may vary.

2.8.2 Prerequisites

You can use pyenv for getting a specific python version.

```
sudo apt-get update; sudo apt-get install make build-essential libssl-dev zlib1g-dev \
libbz2-dev libreadline-dev libsqlite3-dev wget curl llvm \
libncursesw5-dev xz-utils tk-dev libxml2-dev libxmlsec1-dev libffi-dev liblzma-dev

curl https://pyenv.run | bash
```

Append the following to your `~/.bashrc`:

```
export PATH="$HOME/.pyenv/bin:$PATH"
eval "$(pyenv init --path)"
eval "$(pyenv virtualenv-init -)"
```

… and ensure to execute/source this as well (`exec $SHELL`).

Now you can install a specific python version:

```
pyenv install 3.12
pyenv local 3.12
```

Install pipenv:

```
pip install --user pipenv
```

2.8.3 Clone Rsepository

```
git clone git@github.com:bihealth/auto-acmg.git
```

Then you need to pull large files from the LFS:

```
git lfs pull
```

2.8.4 Install Dependencies

You can use the provided `Makefile` files to install the dependencies.

```
make deps
```

2.8.5 Set up the SeqRepo

AutoACMG uses the SeqRepo to fetch the reference sequences. To set up the SeqRepo, you'll need to have the RefSeq reference genomes of versions GRCh37 and GRCh38. You can get them as follows:

1. Install dependencies for SeqRepo: SeqRepo uses tabix and bgzip to index and compress the reference genomes. You can install them using the following command for Linux:

```
sudo apt-get install tabix bgzip
```

and for MacOS:

```
brew install htslib
```

2. Initialize the SeqRepo: First, you'll need to setup the SeqRepo instance. You can do this by running the following command:

```
sudo chown $USER /usr/local/share/seqrepo
sudo mkdir -p /usr/local/share/seqrepo
```

Then you can initialize the SeqRepo instance:

```
pipenv run seqrepo init -i auto-acmg
```

Note: The `-i auto-acmg` flag is used to set the SeqRepo instance name to `auto-acmg`. If you want to use a different default seqrepo directory, you can set the `AUTO_ACMG_SEQREPO_DATA_DIR` environment variable or provide the `-r` flag to the `seqrepo` command.

---

2. Download the reference genomes:

The following command downloads the reference sequences. Note, that it'll take some time.

Important: There might be a bug in the SeqRepo that causes the download to fail (refer to the github issue). If this happens, modify the source code by running the following command:

```
sed -i -e 's/if aliases_cur.fetchone() is not None/if next(aliases_cur, None) is not␣
↪None/' \
<your-path-to-lib>/biocommons/seqrepo/cli.py
```

And the actual download command:

```
pipenv run seqrepo fetch-load -i auto-acmg -n RefSeq NC_000001.10 NC_000002.11 NC_000003.
↪11 NC_000004.11 NC_000005.9 NC_000006.11 NC_000007.13 NC_000008.10 NC_000009.11 NC_
↪000010.10 NC_000011.9 NC_000012.11 NC_000013.10 NC_000014.8 NC_000015.9 NC_000016.9 NC_
↪000017.10 NC_000018.9 NC_000019.9 NC_000020.10 NC_000021.8 NC_000022.10 NC_000023.10␣
↪NC_000024.9 NC_012920.1 NC_000001.11 NC_000002.12 NC_000003.12 NC_000004.12 NC_000005.
↪10 NC_000006.12 NC_000007.14 NC_000008.11 NC_000009.12 NC_000010.11 NC_000011.10 NC_
↪000012.12 NC_000013.11 NC_000014.9 NC_000015.10 NC_000016.10 NC_000017.11 NC_000018.10␣
↪NC_000019.10 NC_000020.11 NC_000021.9 NC_000022.11 NC_000023.11 NC_000024.10 NC_012920.
↪1
```

Note: The above RefSeq identifiers are for all of the chromosomes and the mitochondrial genome. Note, that this is a large (and slow) download and will take some time.

2.8.6 Set up the *.env* file

You need to create a *.env* file in the root of the project. The default settings can be found in the *.env.dev* file. Copy the contents with the following command:

```
cp .env.dev .env
```

Important: You need to set the *AUTO_ACMG_SEQREPO_DATA_DIR* variable in the *.env* file to the path of the SeqRepo instance you created in the previous step. It should look similar to this:

```
AUTO_ACMG_SEQREPO_DATA_DIR=/usr/local/share/seqrepo/auto-acmg
```

2.8.7 Running the CLI

After you have set up the SeqRepo and the *.env* file, you can serve the API. You can use the following commands:

```
make serve
```

Then go to the *http://0.0.0.0:8080/api/v1/docs* to see the API documentation.

2.9 REEV API Reference

This section contains the API reference, which we use for data retrieval.

2.9.1 Annonars

Annonars API client.

src.api.reev.annonars.**ANNONARS_API_BASE_URL = 'http://annonars:8080'**

Annonars API base URL

class src.api.reev.annonars.**AnnonarsClient**(*, *api_base_url: str | None = None*)

Bases: object

_get_variant_from_range(*variant: SeqVar | StrucVar*, *start: int*, *stop: int*) → AnnonarsRangeResponse

Pull all variants within a range.

Parameters

- **variant** (*Union[SeqVar, StrucVar]*) – Sequence or structural variant.
- **start** (*int*) – Start position.
- **stop** (*int*) – Stop position.

Returns

Annonars response.

Return type

AnnonarsRangeResponse

api_base_url

Annonars API base URL

cache

Persistent cache for API responses

client

HTTPX client

get_gene_info(*hgnc_id: str*) → AnnonarsGeneResponse

Get gene information from Annonars.

Parameters

seqvar (*SeqVar*) – Sequence variant.

Returns

Annonars response.

Return type

Any

get_variant_from_range(*variant: SeqVar | StrucVar*, *start: int*, *stop: int*) → AnnonarsCustomRangeResult

A wrapper for _get_variant_from_range that handles ranges larger than 5000.

Parameters

- **variant** (*Union[SeqVar, StrucVar]*) – Sequence or structural variant.
- **start** (*int*) – Start position.
- **stop** (*int*) – Stop position.

Returns

Annonars response.

Return type

AnnonarsRangeResponse

get_variant_info(*seqvar: SeqVar*) → AnnonarsVariantResponse

Get variant information from Annonars.

Parameters

seqvar (*SeqVar*) – Sequence variant.

Returns

Annonars response.

Return type

Any

2.9.2 Dotty

Dotty API client.

src.api.reev.dotty.**DOTTI_API_BASE_URL = 'http://dotty:8080'**

Dotty API base URL

class src.api.reev.dotty.**DottyClient**(*, *api_base_url: str | None = None*)

Bases: object

api_base_url

Dotty API base URL

cache

Persistent cache for API responses

client

HTTPX client

to_spdi(*query: str*, *assembly: GenomeRelease = GenomeRelease.GRCh38*) → DottySpdiResponse | None

Converts a variant to SPDI format.

Parameters

- **query** (*str*) – Variant query
- **assembly** (*GRChAssemblyType*) – Genome assembly

Returns
SPDI format

Return type
dict | None

2.9.3 Mehari

Mehari API client.

src.api.reev.mehari.**MEHARI_API_BASE_URL = 'http://mehari:8080'**
Mehari API base URL

class src.api.reev.mehari.**MehariClient**(*, *api_base_url: str | None = None*)
Bases: object

api_base_url
Mehari API base URL

cache
Persistent cache for API responses

client
HTTPX client

get_gene_transcripts(*hgnc_id: str*, *genome_build: GenomeRelease*) → GeneTranscripts
" Get transcripts for a gene.

Parameters
- **hgnc_id** (*str*) – HGNC gene ID
- **genome_build** (*GenomeRelease*) – Genome build

Returns
Transcripts

Return type
GeneTranscripts

Raises
MehariException – if the request failed

get_seqvar_transcripts(*seqvar: SeqVar*) → TranscriptsSeqVar
Get transcripts for a sequence variant.

Parameters
seqvar (*SeqVar*) – Sequence variant

Returns
Transcripts

Return type
TranscriptsSeqVar

Raises
MehariException – if the request failed

get_strucvar_transcripts(*strucvar: StrucVar*) → TranscriptsStrucVar
Get transcripts for a structural variant.

Parameters
strucvar (*StrucVar*) – Structural variant

Returns
Transcripts

Return type
TranscriptsStrucVar
Raises
MehariException – if the request failed

2.10 API Endpoints

The AutoACMG also provides several endpoints for interacting with the system:

async src.api.internal.api.**predict_seqvar**(*variant_name: str = Query(PydanticUndefined)*, *genome_release: str = Query(GRCh38)*)

Predict the ACMG classification of a sequence variant.
Parameters
- **variant_name** (*str*) – The name or identifier of the sequence variant.
- **genome_release** (*str*) – The genome release version.

Returns
The predicted ACMG classification.
Return type
SeqVarPredictionResponse

async src.api.internal.api.**predict_strucvar**(*variant_name: str = Query(PydanticUndefined)*, *genome_release: str = Query(GRCh38)*, *duplication_tandem: bool = Query(False)*)

Predict the ACMG classification of a structural variant.
Parameters
- **variant_name** (*str*) – The name or identifier of the structural variant.
- **genome_release** (*str*) – The genome release version.
- **duplication_tandem** (*bool*) – The duplication is in tandem and disrupts reading frame and undergoes NMD.

Returns
The predicted ACMG classification.
Return type
StrucVarPredictionResponse

async src.api.internal.api.**resolve_variant**(*variant_name: str = Query(PydanticUndefined)*, *genome_release: str = Query(GRCh38)*)

Resolve a variant to a sequence variant or structural variant.
Parameters
- **variant_name** (*str*) – The name or identifier of the variant.
- **genome_release** (*str*) – The genome release version.

Returns
The resolved variant.
Return type
VariantResolveResponse

PYTHON MODULE INDEX

S

INDEX

Symbols

A

C

D

E

F

G

I

T

U

V