# Comments on the VEP VCF parser code

## Snapshot of the code from 5pm 02/02/2016, commit hash 7601efd

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Issue number** | **Class** | **Line(s)** | **Issue(s)** | **Suggestion(s)** |
| 1 | VepVcf.java  parseVepVcf() | 47 | When file is not found at specified command line location, error is not correctly thrown and the failure occurs on line 51. | Wrap file opening in error handling code so exception is thrown immediately and code does not continue to execute. |
| 2 | VepVcf.java  parseVepVcf() | 58 | Error is thrown when the information in the VCF ##SAMPLE header tag is present in the SampleMetaData class but not in the VCF header. Key not found error. Code execution terminates- error message is informative. | Make changes to SampleMetaData class to allow for missing fields or change the error handling in the VepVcf class to also handle this error (in addition to the null pointer error which handles the case where there is no ##SAMPLE tag in the VCF header. |
| 3.1 | VepVcf.java  parseVepVcf() | 66 | Warning correctly thrown, but downstream exception occurs as code unable to execute in the absence of the ##INFO header and associated metadata. | Throw exception rather than warning as code cannot continue through to successful completion. Alternatively handle error elsewhere in the code if successful completion and output generation is desired. |
| 3.2 | VepVcf.java  parseVepVcf() | 66 | Warning correctly thrown, but downstream exception occurs as code unable to execute in the absence of the ##CSQ key within the ##INFO header. | Throw exception rather than warning as code cannot continue through to successful completion. Alternatively handle error elsewhere in the code if successful completion and output generation is desired. |
| 4 | VepVcf.java  parseVepVcf() | 72 | Where the ##SAMPLE header ##ID field is blank an ArrayIndexOutOfBounds exception occurs downstream, which is not very informative when trying to trace the source of the problem (note that a malformed header exception is not thrown). | Throw informative exception that sample names could not be correctly determined from the VCF file header. |
| 5 | VepVcf.java  parseVepVcf() | 74-79 | Noting that code now does not store non-variant bases nor bases which have failed a filter applied (as opposed to previously where it just did not write out information falling into these categories). | N/A- possible issue with AF calc- follow up on this later. |
| 6 | VepVcf.java  deserialiseVepAnnotation() | 180 | Added back in the limit -1 to the String split() method, which prevents clipping of trailing spaces. | Although we don’t use these fields, without this some headers will not be stored- this could be confusing later on. Suggest an explanatory comment if want to remove it again. |
| 7 | VepVcf.java  parseVepVcf() | 88 | Unchecked cast from Object to String | Could get attribute as String, using null as default value. |
| 8 | Output file | N/A | Protein predicted change without associated NP number. | Remove or associate with NP number. |
| 9 | All code | N/A | Column limit of 100 chars exceeded in places. See Google style guide. | Split long lines of code over two lines. |
| 10 | VCF file | N/A | Entries still duplicated within CSQ field | Cannot fix- setting (creation of a set) therefore still required in code. |
| 11 | VepVcf.java  parseVepVcf() | 105 | Potentially confusing retrieval of alleleNum+1. | Add a comment explaining why i+1- magic number, which could easily be thought of as an error on later code modification and removed. |
| 12 | VepVcf.java  parseVepVcf() | N/A | Method violates the single responsibility statement complicating testing, including writing of test cases, and making maintenance of code later more difficult. | Split out different functions into different methods.  In general, methods should do one thing only- see principles of OO design |
| 13 | VepVcf.java  parseVepVcf() | 114-119 | Noting that the code now will not store genotype information where a call could not be made, where the sample was homozygous for the reference allele or has failed an applied filter (as opposed to previously where it just did not write out information falling into these categories). | N/A |
| 14 | VepVcf.java  parseVepVcf() | 126 | Redundant OR (||) in if statement. Both tests check the same field in the VCF (GT). | Remove one? |
| 15 | VepVcf.java  parseVepVcf() | 130 | Reason for return in case of \* unclear. Only two alleles supported- which is correct as previous exception will remove cases that violate this assumption. | Add comment. Note only two alleles supported in case order of code is later changed so that polyploid exception is not triggered prior to this or polyploid samples are later intended to be supported. |
| 16 | VepVcf.java  parseVepVcf() | 135 | Replication of code from lines 135-166. Reason unclear. | Remove replication or justify. Modularise? |
| 17.1 | VepVcf.java  parseVepVcf() | 51 | This method of retrieving the sample metadata appears to only retrieve the information for the first sample. | This method will need to be fixed in a multisample VCF to accommodate multiple ##SAMPLE header lines (pending final Germline Enrichment pipeline output and delivery of test file to confirm). |
| 17.2 | VepVcf.java  setSampleMetaData() | 189 | Requires testing when the above bug is fixed to ensure it operates correctly. | Pending final Germline Enrichment pipeline output and delivery of test file |
| 18 | VepVcf.java  Java docstring | 23 | Code is not solely the product of this author. Potential issues with audit trail. | Change author to incorporate new additional code author. |
| 19 | VepVcf.java  Java docstring | 23 | Completion of Java docstring required including version number etc. | Complete docstring. |
| 20 | VepVcf.java  parseVepVcf() | 144  154  164 | Stores genotype information but no information on whether variant is HET, HOM etc. at the position. | Linked to issue 16. |
| 21 | VepVcf.java  parseVepVcf() | 143  153  163 | Use of genomeVariant object as key makes querying hash map manually for testing difficult (i.e. using .get() method). | - |
| 22 | Pair.java | N/A | Specific function of class unclear in this context. Why create a pair in this way? | Add documentation and more informative variable names or comments to calling code- it’s not clear why a pair has been created in this way. |
| 23 | VepVcf.java  parseVepVcf() | 167 | Could not create a test case to test correct triggering of this warning as attempts failed earlier tests. | Create test to check correct triggering. |
| 24 | VepVcf.java  parseVepVcf() | 188 | Presence of headers not defined in VepAnnotationObject.java causes IllegalArgumentException. | N/A- code no longer robust to changes in output by the VEP. Hash map generated on line 184 therefore potentially no longer required. |
| 25 | VepAnnotationObject.java | N/A | Note that the toString() method of this class specifies how the CSQ annotations are returned. | Add documentation explaining where can change output (if desired). |
| 26 | VepAnnotationObject.java | 278-312 | Two statements per line. | Each statement should be followed by a line break. |
| 27 | GenomeVariant.java |  | Note that minimal representation method has not been unit tested for correct output as it is assumed it has not been changed from its use in previous projects. | - |
| 28 | Main.java  Java docstring | 17 | Code is not solely the product of this author. Potential issues with audit trail. | Change author to incorporate new additional code author. |
| 29 | Main.java  main() | 55 | Option -K writes out variants with no associated VEP annotation in addition to those which contain NM\_\* transcripts. | Change behaviour if this is not intentional. |
| 30 | Main.java  main() | 78-100 | When file is not found at specified command line location, error is not correctly thrown and the failure occurs on line 80 when the headers are attempted to be read by the htsjdk library’s VCF reader. | Wrap file opening in error handling code so exception is thrown immediately and code does not continue to execute. |
| 31 |  |  |  |  |
| 32 |  |  |  |  |
| 33 |  |  |  |  |
| 34 |  |  |  |  |
| 35 |  |  |  |  |
| 36 |  |  |  |  |
| 37 |  |  |  |  |
| 38 |  |  |  |  |
| 39 |  |  |  |  |
| 40 |  |  |  | - unclear how AF is calculated in the absence of the AF annotation (such as would be expected with the Germline Enrichment pipeline)- pending final Germline Enrichment pipeline output and delivery of test file. |
| 41 |  |  |  |  |
| 42 |  |  |  |  |