6 MutationInfo: a tool to automatically infer chromosomal positions from dbSNP and HGVS genetic variants

Alexandros Kanterakis¹, Eugenia Kartsaki², Marieke Bijlsma¹, Pieter Neerincx¹, Theodora Katsila³, George P. Patrinos³, George Potamias², Morris A. Swertz¹

¹Genomics Coordination Center, Department of Genetics, University Medical Center Groningen, University of Groningen, Groningen, The Netherlands
²Institute of Computer Science, Foundation for Research and Technology Hellas (FORTH), Heraklion, Greece
³University of Patras, School of Health Sciences, Department of Pharmacy, University Campus, Rion, Patras, Greece

Submitted as a chapter to Human Genome Informatics: Translating Genes Into Health
Editors: Darrol Baker, Christophe Lambert and George Patrinos
Publication August 2018
Publisher: Elsevier
https://www.elsevier.com/books/human-genome-informatics/baker/978-0-12-809414-3
Abstract

Today there is a plethora of Locus-Specific Mutation Databases (LSMD) that contain genetic variants for many organisms and for a large range of phenotypes and diseases. This knowledge base is expanding almost daily with the discovery of novel variants in a daily base that have various consequences in human physiology. One of the most widely used standards for reporting new variants is HGVS (Human Genome Variation Society). HGVS describes a variant according to a reference sequence, its position and its nucleotide or amino acid change. However, the lack of established quality standards for reporting genetic variants and the plethora of existing reference sequence systems have led to incidental erroneous or ambiguous reporting of variants in LSMDs or even in published reports. Another source of ambiguity is the use of gene names instead of reference sequences in HGVS variants. These inconsistencies hinder the task of identifying the chromosomal position of a HGVS variant, which is a vital part in confirming its existence in a fully genotyped or sequenced sample. Here we review and evaluate the ability of existing tools to infer the genomic position of variants described in HGVS. We also introduce a new tool, MutationInfo, which combines nine existing tools with a BLAT search in order to exhaustively search for the chromosomal position of HGVS variants. To evaluate MutationInfo we applied it to 1,014 variants from dbSNP, 634 HGVS variants that use sequence accession numbers, and 33,337 HGVS variants that use gene names as reference sequences. Its application to dbSNP variants revealed six problematic variants. For HGVS variants with sequence accession numbers it revealed that 129 (20%) could not be analyzed by any individual tool except for MutationInfo’s pipeline. Finally, its application to HGVS variants with gene names revealed the effect of this erratic practice, since half of them could not be unequivocally assigned to a single chromosomal position.

Link: http://www.epga.gr/MutationInfo/
Source code: https://github.com/kantale/MutationInfo
License: MIT License

6.1 Introduction

A large proportion of scientific publications in the field of genomics focus on discovering novel genomic variations in the human genome and investigating known ones. Moreover, there are hundreds of online databases with known variants. For example, LOVD [13], one of the most popular databases for variants, is installed in 89 different instances reporting more than 10 million variants, whereas dbSNP contains more than 100 million...
validated SNPs. dbSNP assigns a unique ID (rs#) for validated variants, in which the specifics of a variant - like the type of mutation and the position - are unambiguously defined. For new or under-investigation variants, the most common nomenclature for reporting genomic variants and their positions is HGVS. According to HGVS, variations are reported in terms of their relative position and basepair alterations in a reference sequence. This sequence can be a transcript, a chromosome or a database accession number. Official HGVS directives were introduced in 2001 and the last update was published in 2016 [11]. Nevertheless, HGVS-like variation reporting was common even in publications in the 1990s [33].

Although HGVS supports the definition of a variant according to a complete reference genome (for example chr1:12345678A>G) this is rarely used in scientific reports. One reason for this is that a complete reference genome was only introduced years after the HGVS was being used in practice. The hg1 human genome reference was introduced in 2000, as part of the Human Genome Project and has undergone 18 major updates since then. Another reason is that HGVS allows the position to be stated according to the location on the mRNA of a gene transcript. According to HGVS this is called coding location (denoted by 'c:' prefix, for example, NM_001104.3:c.1729C>T) in contrast with the genomic location (denoted by the prefix “g.” for example, NG_013304.2:g.18705C>T). The practical value of reporting the coding location is that it is obvious if a variant is in an exonic or intronic region and sometimes it reveals the effect in the translation (for example a variant in a splice-site). This makes the description of the variant more informative since it provides the exact position of the mRNA where the mutation occurred. Another advantage of the mRNA level notation is that it is shorter. For example, a report that investigates a mutation of the BRCA1 gene is more coherent when stating 'c.211A>G ' -rather than- 'chr17:g.41258474T>C of hg19'.

To address potential confusion and ambiguity stemming from multiple reference sequences that describe the same variant, HGVS nomenclature recommends the use of LRG (Locus Reference Genomic) sequence ID for this purpose. LRGs are a manually curated record of reference sequences that guarantees an un-versioned (i.e. unchanged) coding scheme [20]. Nevertheless, since LRG has not been widely adopted, the HGVS consortium also recommends the use of RefSeq or Ensembl transcript. RefSeq sequences are a collection of genomic DNA, gene transcripts and proteins that exhibit certain qualitative criteria for a variety of well studied organisms maintained by NCBI [4].

---

Ensembl reference sequences are based partly on RefSeq but include additional resources like UniProt as well as manually curated entries\(^5\). It is known that the differences between these two resources have effects on certain analysis pipelines \(^41\) including variant annotation \(^22\). Moreover, as we will present later, certain HGVS analysis tools have preferences for specific sources, so that the choice of a reference sequence source in an HGVS variant can affect the readability and subsequently the significance of the variant.

Today there is an increasing desire to identify already reported variants in existing genotyped or sequenced samples, with a strong driver being the clinical genetics community. An example of a relevant service is Promethease\(^6\), which uses a community curated variant database, SNPedia, to locate variants in a set of genotypes provided by the user \(^4\). These efforts belong to the ‘translational bioinformatics’ field, which tries to apply genetics knowledge in clinical and medical practice \(^26\) in order to bring the vision of personalized medicine closer to reality. These services analyze individual genetic profiles that have been generated either by whole genome genotyping techniques or whole genome sequencing after the application of a variant calling pipeline. In both cases the matching between the individual genetic profile and the set of reported variants available is based on the genetic location. Therefore, the process of identifying the chromosomal location of an HGVS variant is becoming increasingly important in clinical genetics.

In theory this variant identification process should be trivial, since an HGVS variant reports a unique genomic position regardless of the chosen coding scheme. However, recent studies have identified several difficulties that severely impede this process. For example, in one study \(^9\), 26 different laboratories were given tumor samples and asked to locate and report genetic variants. Only 6 laboratories reported all the correct variants in the right HGVS format. Among other discrepancies, 4 laboratories reported incorrect positions of indels, while 1 laboratory reported a correct position but an incorrect reference. In another study \(^28\), the authors examined the validity of reported variants in LSMDs and reported common mistakes; among the most prominent errors were numbering issues on the cDNA level as well as on the genetic level of reported transcripts. For example, although the HGVS directives state that the numbering should start from the translation initiation site, some papers report the position according to the transcription initiation site. Many LSMDs copy this information directly from papers without performing basic quality and uniformity checks. Finally, another study \(^33\) focused on mistakes that involved the incorrect use of the reference sequence in

\(^5\)http://www.ensembl.org/info/genome/genebuild/index.html
\(^6\)https://www.promethease.com/
HGVS. These are the omission of the transcript version or the use of a gene name instead of a transcript name. They also discussed the consequences of these mistakes for research and even in health care. Interestingly, errors on variant reporting can occur even in well-designed studies that have submitted their findings to dbSNP [32].

Various tools have been developed to remedy these issues by performing quality checks on reported variants in HGVS. Perhaps the most prominent tools are Mutalyzer [38], BioCommons/HGVS [15] and Counsyl7; they offer methods for HGVS parsing and conversion. Other tools for variant annotation and clinical effect prediction (e.g. VEP (Variant Effect Predictor) [24], Variation Reporter8 and TransVar [42]) also accept input in HGVS and perform this quality check as a secondary task. In contrast to some well-known annotation tools that do not accept HGVS variants as input like ANNOVAR9 [37], GEMINI [27] and SNPnexus [8]. Additionally, existing databases that list variants in HGVS format are LOVD [13] and ClinVar [19].

Here we assess the ability of HGVS-compatible tools to successfully infer the chromosomal positions of variants. We discuss the limitations of each tool and suggest improvements. Further, we introduce a new tool, MutationInfo, which combines nine different existing tools, databases and online resources in order to perform this task. MutationInfo splits variants into three possible categories: dbSNP variants, HGVS with sequence accession numbers, and HGVS with gene names. For each category it applies a different pipeline. To our knowledge, MutationInfo is the only tool that accepts all three types of variants.

As a test set we use three sets of reported variants, one for each category. The first contained the dbSNP variants reported in the PharmGKB database [25]. The second set contained the HGVS variants reported in the same database that happened to have a sequence accession number as a reference sequence. The third set contained the HGVS variants extracted from a large corpus of published abstracts that contained the word “mutation”. This last set of HGVS variants have a gene name as a reference sequence.

MutationInfo is implemented in Python with a very simple programming interface. It offers a simple function get_info that accepts variants in both dbSNP and HGVS. This format agnostic interface makes MutationInfo ideal for fast genomic location and quality check of a set of variants that belong either in dbSNP or HGVS nomenclature.

7https://github.com/counsyl/hgvs
6.2 Existing tools for resolving HGVS position

In this section we describe how to use existing online services and tools that offer position resolution of HGVS variants. All these tools offer position resolution as a small part of their functionality, since there is no tool solely dedicated to this purpose. This section does not cover the many other useful tasks that these tools also perform. Nor do we focus on the ability of these tools to process the complex and uncommon mutation types that HGVS supports.

Being able to automatically resolve positions for HGVS variants through a scripting language relieves us of much repetitive manual work and also contributes to the reproducibility of our analysis. We will use the Python programming language (version 3) and we also assume that the packages requests and pandas are installed. Requests [30] is a package that simplifies access to online resources and pandas [23] offers convenient methods for managing tabular data. We will start by importing these packages:

```python
import requests
import pandas as pd
```

### 6.2.1 Mutalyzer

Mutalyzer [38] offers an online API (Application Programming Interface) to automatically access most of its functionality. Assuming that we want to infer the chromosomal position of the variant NM_001276506.1:c.204C>T in version hg38 of the human genome, we will type:

```python
r = requests.get('https://mutalyzer.nl/json/numberConversion',
                 {
                 'build': 'hg38',
                 'variant': 'NM_001276506.1:c.204C>T'
                 })
data = r.json()
print (data)
OUTPUT: ['NC_000011.10:g.112088901C>T']
```

Unfortunately Mutalyzer offers this functionality only in HGVS variants that are using NM_, NC_, or LRG_ accession numbers. For variants that are using other NCBI reference sequences, Mutalyzer can still be useful by performing a “c. to g.” conversion. For example assuming the variant NG_008377.1.c.-1289G>A, we can apply the following commands:

```python
r = requests.get('https://mutalyzer.nl/json/numberConversion',
                 {
                 'build': 'hg38',
                 'variant': 'NG_008377.1:c.-1289G>A'
                 })
data = r.json()
print (data)
OUTPUT: ['NC_000011.10:g.112088901C>T']
```
Knowing the genomic position of a variant can help us to recover the chromosomal position, as we will demonstrate later.

6.2.2 The HGVS python package

The HGVS Python package [15] offers methods to parse variants, perform “c. to g.” conversions and map a variant to arbitrary transcripts. To perform these operations HGVS maintains a connection with a database that contains transcript information. By default this database is an instance of UTA (Universal Transcript Archive). To perform position resolution with HGVS, we initially have to import the appropriate HGVS modules:

```python
import hgvs.parser
import hgvs.dataprotviders.uta
import hgvs.assemblymapper

hdp = hgvs.dataprotviders.uta.connect()
hgvs_parser = hgvs.parser.Parser().parse_hgvs_variant
assembly_mapper = hgvs.assemblymapper.AssemblyMapper()
transcript_mapper = hgvs.variantmapper.VariantMapper(hdp)
```

In this example, the object `hgvs_parser` validates and parses HGVS variants, the object `assembly_mapper` maps a variant to a reference genome and the object `transcript_mapper` maps a variant to another transcript. To locate the chromosomal position of the variant `NM_001276506.1:c.204C>T` that we also used in the Mutalyzer example we will have to use the following commands:

```python
var = 'NM_001276506.1:c.204C>T'
var_parsed = hgvs_parser(var)
var_mapped = assembly_mapper.c_to_g(var_parsed)
```
As we can see, the result is the same as the one obtained from Mutalyzer. Mutalyzer managed to perform “c. to g.” conversion for variant NG_008377.1:c.-1289G>A, but failed to locate a chromosomal position. Interestingly, HGVS shows an inverse functionality: it failed to perform “c. to g.” on this variant, but succeeded in locating a chromosomal position. For this reason we will take the output of the “c. to g.” operation from Mutalyzer, which is NG_008377.1:g.5165G>A and we will locate a chromosomal position with HGVS:

```python
var = 'NG_008377.1:g.5165G>A'
var_parsed = hgvs_parser(var)
transcripts = assembly_mapper.relevant_transcripts(var_parsed)
print (transcripts)
OUTPUT: ['NM_000762.5']

var_NM = transcript_mapper.g_to_c(var_parsed, 'NM_000762.5')
print (var_NM)
OUTPUT: NM_000762.5:c.144G>A

print (assembly_mapper.c_to_g(var_NM))
OUTPUT: NC_000019.10:g.40850283C>T
```

This is also a demonstration of the synergy that can be achieved by combining two or more tools.

### 6.2.3 Variant Effect Predictor

The primary use of Variant Effect Predictor (VEP) [24] is to annotate a variant with effect information (for example consequences, SIFT and PolyPhen scores). Additionally, VEP also provides useful location information, including possibly affected transcripts. Since VEP belongs to EMBL’s large family of bioinformatics tools, it also supports Ensembl transcripts. Accessing VEP is easy through the `requests` library. Initially, we define VEP’s URL and the required HTML headers:

```python
headers={ "Content-Type" : "application/json"}
vep_url = "https://rest.ensembl.org/vep/human/hgvs/{var}?"

def vep(var):
    url = vep_url.format(var=var)
    response = requests.get(url, headers=headers)
    if response.ok:
        return response.json()
```

164
Now we can request location information for a specific HGVS variant:

```python
var = 'NM_001276506.1:c.204C>T'
data = vep(var)
print (data[0]['assembly_name'])
OUTPUT: GRCh38
print (data[0]['seq_region_name'])
OUTPUT: 11
print (data[0]['start'])
OUTPUT: 112088901
```

Again we notice that the output has the same information as the previous methods. VEP failed to locate position information for both the NG_008377.1:c.-1289G>A and NG_008377.1:g.5165G>A variants.

### 6.2.4 Variation reporter

Variation Reporter belongs to the NCBI family of bioinformatics tools and provides a similar functionality to VEP. As with VEP, it focuses on providing the clinical effects of variants and it also resolves positions. On the following lines, we define the function `Variation_Report`er, which handles the requests to this service.

```python
def Variation_Report(var):
    r = requests.post(url, {'annot1': var, })
    if r.ok:
        return r.text
    else:
        print ('ERROR:')
        print (r.text)
```

Variation Reporter generates data in simple text format as opposed to other tools that export data in a structured JSON format. This means that users have to write custom functions to parse the output. The following listing is a simple function that converts output from Variation Reporter into a pandas object [23].
from io import StringIO

def parse_vr_results(results):
    lines = results.split('\n')
    lines = filter(lambda x : x[0:2] not in
       ['Su', '..', '...', '##'], lines)
    lines = '\n'.join(lines)

    data_f = StringIO(lines)
    return pd.read_csv(data_f, sep='\t')

We can demonstrate the functionality of the above with the following commands:

var = 'NM_001276506.1:c.204C>T'
vr_results = Variation_Reporter(var)
vr_parsed = parse_vr_results(vr_results)

print (vr_parsed['Hgvs_g'][0])
OUTPUT: 'NC_000011.9:g.111959625C>T'

We notice that Variation Reporter mapped the variant in an older genome assembly (hg19) compared to other tools. Fortunately, apart from VEP, the EMBL also includes a service that converts the coordinates of a variant from one assembly to another. This tool is Assembly Converter and can be accessed through the EMBL’s REST API with the following code:

AC_url = 'https://rest.ensembl.org/map/human/GRCh37'
AC_url += '/(chr):{pos}..{pos}:1/GRCh38'

def from_19_to_38(chromosome, position):
    url = AC_url.format(
        chr=chromosome,
        pos=position
    )
    headers = {'content-type':'application/json'}
    request = requests.get(url, headers=headers)

    if request.ok:
        data = request.json()
        return data['mappings'][0]['mapped']['start']
    else:
        print ('ERROR:')
Now we can convert the variant NC_000011.9:g.111959625C>T from hg19 to hg38:

```python
new_location = from_19_to_38(11, 111959625)
pd.DataFrame(new_location)
OUTPUT: 112088901
```

Again we notice that the location is the same as the one inferred from the previous tools. In addition, Variation Reporter succeeded in locating a chromosomal position for NG_008377.1:g.5165G>A but failed for NG_008377.1:c.-1289G>A. Converting a genomic locus from one assembly to another is called “liftover”. NCBI also has a tool for this task, which is called Remap. Remap is available either through a web interface or through a Perl script.

### 6.2.5 TransVar

TransVar [42] is a tool that applies various annotations and conversions for genomic loci that are defined in HGVS. Assuming a common Linux or OSX command line environment, TransVar can be installed with the following commands:

```bash
pip install transvar
pip install pytabix
transvar config --download_anno --refversion hg38
transvar config --download_ref --refversion hg38
```

The last two commands download a set of annotation databases locally that TransVar consults. These databases include RefSeq (NCBI), Ensembl, UCSC and AceView [36]. TransVar is a command-line tool, which means that we cannot access its functionality through Python. Nevertheless, we can execute the tool from the command line and then process the output. For example the following commands consult the RefSeq database for the ‘NM_001276506.1:c.204C>T’ variant:

```bash
transvar canno -i 'NM_001276506.1:c.204' --refseq --refversion hg38 > output.txt
```

```python
data = pd.read_csv('output.txt', sep='\t')
pd.DataFrame(data['coordinates(gDNA/cDNA/protein)'][0])
OUTPUT: chr11:g.112088901C/c.204C/.
```

Again we can verify that this is the same position that was inferred with all the previous tools. TransVar failed to process variants having the sequence NG_008377.1, since it could not locate it in any of the databases that it supports.
6.2.6 BioPython

BioPython [5] is Python’s best known library for biological computation. It contains a large variety of methods for accessing online genetic databases, sequence annotation, statistical inference, machine learning and clustering. Here, we will show how we can use BioPython to infer the chromosomal position of the HGVS variant: NG_008377.1:g.5165G>A. Initially, we will download the reference sequence (NG_008377.1) in FASTA format and then we will apply BLAT in order to identify where this sequence is aligned in the human genome. Assuming we have BioPython installed, initially we import the necessary libraries:

```python
from Bio import Entrez
from Bio.Blast import NCBIWWW
from Bio.Blast import NCBIXML
Entrez.email = 'anonymous@gmail.com'
```

The Entrez library [21] allows access to NCBI’s large collection of databases. With the following commands, we download the reference sequence NG_008377.1 of the variant in FASTA format:

```python
handle = Entrez.efetch(
    db='nuccore',
    id='NG_008377.1',
    retmode='text',
    rettype='fasta')
fasta_data = handle.read()
```

We can inspect the `fasta_data` variable to confirm that it contains a FASTA record. The next step is to perform a BLAT search. BioPython offers many options for the BLAT algorithm, target database and organisms. A full list of available target databases for NCBI is available in NCBI’s FTP site documentation [34]. The following commands perform a BLAT search in the human reference genome and then parse the results.

```python
result_handle = NCBIWWW.qblast("blastn", "refseq_genomic_human", fasta_data)
blast_record = NCBIXML.read(result_handle)
```

A BLAT search results in various alignments whose number and quality depend on the choice of target database. These alignments are usually called “hits”. Each alignment has many HSPs (high-scoring pairs) which are fragments (or frames) of queried-reference sequence pairs accompanied by statistics produced from the search algorithm. For our example we will take the first HSP from the first alignment of the BLAT search:

```python
first_sequence = blast_record.alignments[0]
```
Here we have the chromosome and reference genome information. Also, the “expectation” is a metric of the quality of the match. The 0.0 value of the “expectation” in our case indicates a perfect match. With the following commands we get the absolute position of the aligned sequence in the reference genome:

```python
print (first_match.sbjct_start)
OUTPUT: 40855447
print (first_match.sbjct_end)
OUTPUT: 40841538
print (first_match.query_start)
OUTPUT: 1
```

Here we notice that the start is lower than the end. This indicates that the sequence that we queried was aligned in the negative strand of the reference. Therefore, in order to locate the position of the NG_008377.1:g.5165G>A variant, we need to count back 5,165 nucleotides before the start position. If the alignment was in the positive strand, then we should count forward (plus). The final calculations are as follows:

```python
result = first_match.query_start + first_match.sbjct_start - 5165
print (result)
OUTPUT: 40850283
```

For validation, we confirm that the result is the same as the one obtained from the HGVS Python package.

### 6.3 Methods

From the previous section, it is evident that existing tools vary in their ability to resolve positions of HGVS variants. Moreover, this task often requires the combination of two or more tools. To alleviate this problem we present MutationInfo, which is primarily a frontend to access existing tools for recovering the chromosomal position of HGVS and
dbSNP formatted variants. It offers a simple function named `get_info`, which accepts variants in both formats. The methods are summarized in Figure 6.1.

If a variant with a dbSNP name is detected (the format should be ‘-rs#’) then MutationInfo applies the following pipeline: it queries Variant Effect Predictor (VEP), MyVariant.info \[39\], \[40\] and CruzDB \[29\] in this order. If any of them succeeds in recovering a chromosomal position then it returns this position and stops, otherwise it moves to the next service in the list. The order of the tool list was constructed according to their efficiency to locate variants in PharmGKB. VEP is a service from Ensembl that determines the effect of a variant. It is basically an aggregator of useful tools like SIFT \[18\] and PolyPhen \[1\], along with basic location and reference/alternative information. Although VEP also accepts variants in HGVS, it does not recognize reference accession numbers from RefSeq (it recognizes gene names and Ensembl transcripts). Since the vast majority of reported HGVS variants are using a RefSeq accession number, MutationInfo does not use this tool for HGVS processing in the default settings. The second tool is MyVariant.info, which is a variant annotation service that has collected information from various resources\(^{10}\). The documentation of this tool \[39\] states that it offers variant annotation for HGVS variants that state explicitly the chromosomal position (i.e. chr1:g.35367G>A). Although it mentions that alternative reference sequences (like NM_003002.2:c.274G>T) are also supported, we could not verify this by running the tool. For this reason MutationInfo by default uses this tool only for dbSNP variants. Finally CruzDB \[29\] is a Python interface for the database tables of UCSC’s genome browser and it does not support HGVS.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Erroneous</th>
<th>Correct</th>
<th>PharmGKB cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;-&gt;&quot; instead of &quot;;-*&quot;</td>
<td>1048G-&gt;C</td>
<td>1048G&gt;C</td>
<td>121</td>
</tr>
<tr>
<td>Multiple alternatives</td>
<td>844G&gt;C/T</td>
<td>844G&gt;C, 844G&gt;T</td>
<td>8</td>
</tr>
<tr>
<td>Parentheses</td>
<td>-1126(C&gt;T)</td>
<td>-1126C&gt;T</td>
<td>75</td>
</tr>
<tr>
<td>Invalid substitution</td>
<td>1160CC&gt;GT</td>
<td>1160_1161delinsGT</td>
<td>2</td>
</tr>
<tr>
<td>Total: 206</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.1: Parsing problems of HGVS variants in PharmGKB. MutationInfo corrects these problems.

If a variant is not in dbSNP format (‘rs#’), then MutationInfo assumes that it is

\(^{10}\)For a complete list see: \[http://myvariant.info/metadata\]
formatted as HGVS. Initially, it checks if the variant contains any common format error. MutationInfo recognizes and corrects 4 common mistakes in HGVS formatting (see Table 6.1) that made all the available tools fail. Once potential format errors have been corrected, MutationInfo tries to identify if the reference sequence is a sequence accession number or a gene name. Depending on this result, it follows two different pipelines presented on the following sections.

### 6.3.1 Pipeline for HGVS variants with gene names

MutationInfo detects when a reference sequence is a gene name. To accomplish this, it attempts to locate the provided reference sequence in the HGNC gene name nomenclature. The HUGO Gene Nomenclature Committee (HGNC) [14] maintains a list of officially recognized gene names, accompanied by synonyms and previously used names. If the reference sequence matches a gene name (or a synonym), it applies the following method: it attempts to run the TransVar tool and to also access the VEP online service 11 in order to determine a chromosomal position. If both fail, or if inconsistencies between them are detected, it reports a relevant error message to the user.

In terms of efficiency TransVar has two advantages over VEP: the first is that it is available as both a web service and as a locally installed tool, which allows us to perform thousands of queries without being concerned about a possible service disruption. The second is that TransVar accepts HGVS notations that do not include the alteration part (this is the sequence change, i.e. A>G). When this happens, TransVar returns all the possible reference sequences and chromosomal locations that match this specific notation. For example, the HGVS **BRCA1:c.5251** returns three possible transcripts, which all match three different positions and all have three different reference nucleotides (C, A and T). This multitude of positions is due to the existence of two alternative splices for this particular coding location. In contrast, the **BRCA1:c.5251C>T** returns only the reference sequence that has the nucleotide ‘C’ in this location. Knowing about possible alternative splices in the location of a variant can be extremely important for understanding its functional consequences, as shown in cancer studies [16]. One disadvantage of TransVar is that it derives the chromosomal position of HGVS variants with gene names only for the hg19 version of the genome assembly.

---

11 VEP, Fetch variant consequences for multiple HGVS notations: https://rest.ensembl.org/documentation/info/vep_hgvs_post
6.3.2 Pipeline for HGVS variants with sequence accession numbers

If MutationInfo detects a sequence accession number then it runs a pipeline for the purpose of recovering a chromosomal position for the variant. The first step is to use the ‘spline’ and ‘blat’ method of BioCommons [15]. If both of these methods fail, then it tries the Counsyl tool. Although these tools do not have the highest efficiency, according to our experiments, they do perform the complete computation locally without requiring Internet access to external services. Therefore, for the sake of speed, MutationInfo prioritizes use of these tools. If these tools fail, MutationInfo makes a request to Mutalyzer’s Position Converter tool\(^\text{12}\). This is a tool dedicated to locating a chromosomal position in HGVS variants. Position Converter usually fails if the HGVS reference sequence is not an NM_ RefSeq transcript. However, the Name Checker tool in Mutalyzer, although its purpose is not solely to locate a chromosomal position, is very successful in making ‘c. to g.’ conversions. This is the task of converting the position of a mutation from coding DNA to genomic coordinates\(^\text{13}\). Whenever Position Converter fails then MutationInfo makes a request to the Name Checker tool from Mutalyzer. This task can have three possible outcomes: the first is that Name Checker recovers the chromosomal position of the variant and performs a ‘c. to g.’ conversion; the second is that it succeeds only in the ‘c. to g.’ conversion; and the third is that it returns an error. If the first scenario happens, then MutationInfo returns the chromosomal position and stops. For the second MutationInfo takes the genomic coordinates and performs a BLAT search to find the chromosomal position. For the third scenario, MutationInfo downloads the reference sequence’s GenBank record from NCBI. The GenBank record has information on the relative genomic positions of the CDS (if any) and the start and end positions of the sequence’s exons (if any). MutationInfo feeds this information to a BioPython \(^\text{14}\) external module that performs a ‘c. to g.’ conversion. Subsequently, based on the genomic position of the variant, MutationInfo performs a BLAT search to recover the chromosomal position.

Whenever needed, the BLAT method in MutationInfo works as follows: MutationInfo downloads the FASTA file of the reference sequence of the variant. Given that the genomic position (g.) of the variant is known, it first checks if the reference sequence in FASTA is the same as the reference in the HGVS variant. The next step is to isolate a sub-sequence starting 20,000 nucleotides upstream to 20,000 downstream from that genomic position (or shorter if the start or end of the reference sequence is reached).

\(^{12}\)https://mutalyzer.nl/position-converter

\(^{13}\)http://www.hgvs.org/mutnomen/recs.html#general

\(^{14}\)In GIT terms, this is a ‘fork’ of BioPython that has not been ‘merged’ to the official release:
https://github.com/lennax/biopython/tree/f_loc5/Bio/SeqUtils/Mapper
Subsequently it accesses UCSC’s BLAT search service [17], [3] and performs a sequence alignment of the isolated sequence in the human reference genome. Finally, it reports the position with the highest BLAT score [3].

Figure 6.1: MutationInfo’s default pipeline for recovering the chromosomal position of a genetic variant.

Figure 6.1 shows the default pipeline that MutationInfo follows to recover the chromosomal position of a genetic variant. For more customized tasks, the user can provide additional options to use only specific tools. One limitation of the current implementation is that although the desired genome assembly is an optional parameter, MutationInfo does not guarantee that the returned position will belong on this assembly. The reason for this is that certain tools are still not fully adapted to the hg38 version, therefore an older assembly is used. Of course the returned item always contain the version of the genome assembly used. In our future work we plan to embed liftover methods in order to offer seamless conversion between genome assemblies.
Finally some additional implementation notes: MutationInfo keeps a local storage of all data generated from requests on external services in order to minimize Internet access and possible disruption to these services from repetitive calls that contain common variants. Moreover, it includes a setup script, which installs all software dependencies (BioCommons, Counsyl, CruzDB, BioPython) along with their required datasets (i.e. reference genomes). Finally, MutationInfo contains a web application implemented with the Django web framework\(^\text{15}\) that is easily embeddable in other web services. The purpose of this application is to offer a lightweight web service that infers (or validates) the chromosomal position of HGVS (or dbSNP) variants as an extra component in existing LSMD installations.

### 6.4 PharmGKB as a testing platform

To evaluate the ability of MutationInfo to locate the chromosomal positions of dbSNP and HGVS variants with sequence accession numbers, we analyzed the highly curated content of PharmGKB \[25\].

PharmGKB \[25\] is a prominent database in the field of pharmacogenetics. The four main benefits of using PharmGKB as a testing platform are (1) it contains variants with direct clinical significance, (2) it does not have any standardized method for variant reporting as it contains variants in both dbSNP, HGVS and arbitrary formats, (3) it is manually curated, and (4) it contains variants published over a long time (1995 onwards). These characteristics make PharmGKB both very important as an LSDB and, at the same time, error-prone with regard to the quality and validity of the variants it contains. The evaluation of PharmGKB variants from MutationInfo revealed that certain classes of HGVS variants could not be correctly located in a reference genome by existing tools. Consequently, we applied MutationInfo as an alternatively pipeline that attempts to use a variety of existing tools and if they all fail then a BLAT search is used. Using this pipeline we located the genetic location of all HGVS variants in PharmGKB. Additionally, we identified errors that most probably came from careless transfer of variants from papers to the database. We also identified errors that could be attributed to the incorrect use of HGVS notation in the original papers.

PharmGKB is an LSMD with variants and haplotypes of pharmacogenomics interest that are collected manually from publications. The content of PharmGKB can be divided into 4 categories: Variant annotations, Clinical annotations, Dosing guidelines, and Haplotypes. The difference between clinical annotations and dosing guidelines is that the latter are supported by official or professional societies, whereas the former

\(^{15}\)https://www.djangoproject.com/
contain information with lower scientific support. Both are referred to either in individual variants or in haplotypes that contain many variants. Although the clinical interpretation of single variants is relatively straightforward, haplotypes are more challenging since they are continuously updated [7]. Since haplotypes contain more dynamic and thus error-prone content, in this study we focused on variants that comprise the haplotypes listed in PharmGKB.

PharmGKB’s website \(^{16}\) was accessed in May 2016. It contained 89 haplotypes from 78 different genes \(^{17}\). These haplotypes contained 1,554 different variants in total. 880 were in dbSNP format, 634 are in HGVS format and 40 were free text descriptions of the variants (e.g. ‘TAA ins between 1065 and 1090’). The reference sequences were not directly available for 542 of these HGVS variants. To locate this we used either the reference sequence suggested by the official nomenclature of the gene (e.g. cypalleles for the CYP gene family), or the more frequently occurring transcript of the gene according to NCBI’s gene search service \(^{18}\). For our analysis we tested all the tools available in MutationInfo’s pipeline (VEP, MyVariant.info, CruzDB, BioCommons, Counsyl, Mutalyzer, BLAT) along with 2 external tools (Variant Reporter, TransVar) and 1 database (LOVD). We used the dbSNP and HGVS formatted variants for the evaluation, (described below), and excluded the free text descriptions from our analysis.

\(^{16}\)https://www.pharmgkb.org/

\(^{17}\)https://www.pharmgkb.org/search/browse.action?browseKey=haplotypeGenes

6.4.1 Analysis of dbSNP variants

PharmGKB contained 1014 SNPs in RS# format of which 880 were unique. All three tools were able to infer a genetic position for 781 SNPs, 91 SNPs were located only by VEP and CruzDB, 6 SNPs were located only by VEP, and 1 SNP was located only by MyVariant.info. There were no SNP located only by CruzDB (see Venn diagram in Figure 6.2). Finally, one SNP (rs2839940) was not identified by any of the available tools. After closer examination it was revealed to be an erroneously reported SNP in PharmGKB, the correct name was rs28399440. We also located the following problems: rs72549375 from CYP1B1 does not exist in dbSNP although it is referred to in the relevant cypalleles page; rs35044222, rs67944833, rs67878449 from SULT1A3 gene are listed in dbSNP with no position information; and rs72558184 from CYP2C9 does not exist in cypalleles but is referred to in PharmGKB. This SNP does not have position information in dbSNP. For these 5 SNPs although the dbSNP ID (rs#) was declared in PharmGKB we used instead the HGVS notation that was also available.

All tools (CruzDB, VEP, MyVariant.info) returned the same location for the same SNPs. Nevertheless we noticed that for indels, the tools returned different start and end positions. Although these values were consistent for each tool, the semantics of ‘start’ and ‘end’ positions differed. In table 6.2, we present an example of the different start and end positions returned by all tools for an insertion and a deletion. For insertions, CruzDB returned the same position as start and end. In contrast, in VEP and in...
<table>
<thead>
<tr>
<th>Type</th>
<th>Insertion</th>
<th>Deletion</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs#</td>
<td>rs72549385</td>
<td>rs3215983</td>
</tr>
<tr>
<td>HGVS</td>
<td>NC_000002.12:g.38074887_38074888insA</td>
<td>NC_000001.11:g.46815075_46815076delAT</td>
</tr>
<tr>
<td></td>
<td>Start</td>
<td>End</td>
</tr>
<tr>
<td>CruzDB</td>
<td>38074887</td>
<td>38074887</td>
</tr>
<tr>
<td>VEP</td>
<td>38074888</td>
<td>38074887</td>
</tr>
<tr>
<td>MyVariant.info</td>
<td>38074887</td>
<td>38074888</td>
</tr>
</tbody>
</table>

Table 6.2: Differences in start and end positions returned by different tools for insertions and deletions.

MyVariant.info, the start position differed from the end position by the length of the insertion. Surprisingly, VEP’s start position was higher than the end position. For deletions, all tools returned the same end position. Nevertheless, the start position of CruzDB was lower than the end position by a value equal to the size of the deletion. In general, the only tool that provided values in accordance with the HGVS equivalent of the dbSNP variant was MyVariant.info. These differences can confound an analysis pipeline or researcher that uses more than one tool. MutationInfo returns consistent location information regardless of the underlying tool that has been used. This location is the same with the HGVS start position as if a chromosomal reference sequence had been used (e.g. chr1:123456_123457insA).

### 6.4.2 Analysis of HGVS variants with sequence accession numbers

PharmGKB contains 634 variants in HGVS format. Below we summarize the performance of various tools to identify the correct position and base changes for different kind of reference sequences. Table 6.3 contains the number of variants in PharmGKB that were correctly located as well as the mistakes that were found during this procedure.
<table>
<thead>
<tr>
<th>Method</th>
<th>NT_</th>
<th>NG_</th>
<th>NM_</th>
<th>NC_</th>
<th>GenBank</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total in PharmGKB</td>
<td>118</td>
<td>187</td>
<td>207</td>
<td>59</td>
<td>63</td>
<td>634</td>
</tr>
<tr>
<td>Mutalyzer</td>
<td>0</td>
<td>0</td>
<td>207</td>
<td>0</td>
<td>0</td>
<td>207</td>
</tr>
<tr>
<td>Mutalyzer+BLAT</td>
<td>116</td>
<td>183</td>
<td>0</td>
<td>0</td>
<td>21</td>
<td>320</td>
</tr>
<tr>
<td>Mutalyzer+BLAT after error correction</td>
<td>117</td>
<td>185</td>
<td>0</td>
<td>0</td>
<td>21</td>
<td>323</td>
</tr>
<tr>
<td>Biocommons/Splign</td>
<td>0</td>
<td>124</td>
<td>203</td>
<td>59</td>
<td>0</td>
<td>386</td>
</tr>
<tr>
<td>Biocommons/Blat</td>
<td>0</td>
<td>115</td>
<td>40</td>
<td>59</td>
<td>0</td>
<td>214</td>
</tr>
<tr>
<td>Counsyl</td>
<td>0</td>
<td>0</td>
<td>207</td>
<td>0</td>
<td>0</td>
<td>207</td>
</tr>
<tr>
<td>LOVD</td>
<td>0</td>
<td>0</td>
<td>155</td>
<td>0</td>
<td>0</td>
<td>155</td>
</tr>
<tr>
<td>MutationInfo's BLAT</td>
<td>0</td>
<td>172</td>
<td>203</td>
<td>59</td>
<td>42</td>
<td>476</td>
</tr>
<tr>
<td>Variation Reporter</td>
<td>0</td>
<td>143</td>
<td>205</td>
<td>59</td>
<td>1</td>
<td>408</td>
</tr>
<tr>
<td>TransVar</td>
<td>90</td>
<td>7</td>
<td>203</td>
<td>0</td>
<td>3</td>
<td>303</td>
</tr>
<tr>
<td>Mistakes found</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>Mistakes corrected</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 6.3: Summary of the number of HGVS variants from PharmGKB processed correctly by various tools to infer a chromosomal position.

NT_ transcripts represent genomic sequences that have been generated through a scaffold construction technique from a set of read contigs (usually through a Whole Genome Shotgun sequencing technique). PharmGKB contains 118 variants of this kind, which all belong to the UGT1A gene family. The SNPs and haplotype cataloging of this gene family is maintained by the UGT official nomenclature. According to this nomenclature, all variants are using the NT_005120.15 reference sequence, which is a genomic contig for chromosome 2 and thus includes numerous genes. Therefore, the inclusion of the gene name in the HGVS names of these variants is a prerequisite for their correct description. The success of tools to correctly identify these variants is crucial for accurate interpretation of genetic data.

https://www.pharmacogenomics.pha.ulaval.ca/ugt-alleles-nomenclature/
varied. Mutalyzer supports the inclusion of the gene name in the reference field of HGVS (e.g. NT_005120.15(UGT1A9):c.8G>A). Although there isn’t any explicit rule against the use of parentheses in the reference sequence according to the official HGVS guidelines, all other tools (BioCommons and Counsyl) fail to parse this form successfully. TransVar and Variation Reporter also failed to recognize it. Moreover, Mutalyzer reports the genomic position of these variants in the transcript or else performs a “c. to g.” conversion but fails to report a position in the reference genome. In order to complete this task MutationInfo performed a BLAT search. We were able to locate chromosomal positions for 116 out of 118 variants with this pipeline. After closer examination we found that one variant NT_005120.15 (UGT1A1):c.-1353C>A was copied erroneously from the UGT official nomenclature page and the correct term is NT_005120.15 (UGT1A1):c.-1353A>C (reverted reference / alternative). The second error, variant NT_005120.15(UGT1A1):c.872C>T was reported to have an incorrect reference nucleotide (C instead of A) and we could not correct it. Finally, when we replaced the reference of the NT_ variants with the gene name (e.g. UGT1A9:c.8G>A), TransVar was able to locate a genomic position for 90 out of 118 NT_ variants.

Sequences with the NG_ prefix describe incomplete genomic regions. PharmGKB contains 187 HGVS variants of this kind, all of which belong to the CYP gene family. According to the cypalleles page\(^2\), which is the official allele nomenclature for Cytochrome P450 enzymes, variants of CYP1A2, CYP2A6, CYP2A13, CYP2B6, CYP2C8, CYP3A4 and CYP3A5 genes are using an NG_ transcript for a sequence reference. Mutalyzer successfully parsed and converted these to genomic coordinates for 183 out of 187 variants, but it was unable to locate a position in the reference genome. For this purpose MutationInfo performed a BLAT search (similar to the NT_ variants). During manual inspection of the failed variants we found four mistakes. Two of them belonged to the CYP1A2 gene. Specifically, -3594T>G was erroneously reported as -3954T>G and -3113G>A was reported as -3113A>G. These errors were corrected. The third error included the variant 6999T>C of the CYP2A6 gene [2], which had an incorrect reference, and the 1630T>C variant of the same gene, which was reported twice (same position) but with different reference sequences. These problems were not corrected. Other tools performed significantly worse in identifying genomic positions. BioCommons’ splign method found positions for 124 variants, the BLAT method in the same tool found positions for 115, while the Counsyl tool did not find any positions. Of the external tools, Variation Reporter found positions for 143 variants, whereas TransVar found only 7.

PharmGKB contains 207 HGVS variants where the reference sequence is an mRNA

\(^2\)The Human Cytochrome P450 (CYP) Allele Nomenclature Database: http://cypalleles.ki.se/
transcript with the -“NM_”- prefix. These transcripts are “mature” and do not contain intronic sequences\textsuperscript{21}. Moreover they are qualitatively curated and carefully selected coding transcripts for the human genome (as well as other organisms) from RefSeq\textsuperscript{22}. Most genetic studies that examine variations in transcribed mRNA are reporting variants in this reference format, and most tools for HGVS variant parsing and analysis focus on this format. This is evident from the fact that both Mutalyzer and Counsyl were able to locate a chromosomal position for all 207 variants. Biocommons’ splign method found positions for 203 variants and the BLAT method in the same tool found positions for 40. Of the external tools, Variation Reporter found positions for 205 variants and TransVar for 203. Interestingly, only 155 of these variants were located in LOVD. During this process we found 4 problematic variants. These were three variants from CYP2E1 (NM\textunderscore 000773.3:c\textunderscore .-333A\textless{}T, NM\textunderscore 000773.3:c\textunderscore .-71G\textless{}T, NM\textunderscore 000773.3:c\textunderscore .-352A\textless{}G) and 1 variant from TPMT (NM\textunderscore 000367.2:c\textunderscore .-178C\textless{}T). In all four variants the declared nucleotide position appears to precede the start of the transcript. This constitutes an error for tools that are limiting their regions on the available transcripts. For example, the Name Checker tool from Mutalyzer reported an error but the Position Converter tool, from the same service, succeeded in locating a chromosomal position. Moreover, the BLAT method from MutationInfo failed for these variants since the ‘c. to g.’ conversion yielded a negative value.

A reference sequence in an HGVS variant that has an NC\_ accession type uses a complete genomic molecule, usually a reference assembly. PharmGKB has 59 variants of this kind. These are the easiest types for finding a chromosomal position since the position in HGVS is by definition the position on the chromosome. Although these variants do not have to be processed by any tool to locate their chromosomal position, we found that some tools failed. Mutalyzer and Counsyl, for example, reported errors. Mutalyzer complained that the size of the reference exceeds the threshold of 10 Megabytes. BioCommons’ splign and BLAT methods were able to find positions for 6 out of 59 variants, with these 6 variants belonging to the hg19 reference assembly, whereas the other 53 belonged to the hg38 assembly. BioCommons supported hg38 in its latest version and by updating it we were able to perform conversions for all variants. MutationInfo’s BLAT method also identified correctly the positions for all variants. Although this method had to perform the seemingly unnecessary task of downloading the complete FASTA files for the chromosomes and then perform a BLAT search, it also confirmed that the reference in the HGVS variants were correct and it therefore offered an extra quality check.

\textsuperscript{21}Also see: http://www.hgvs.org/mutnomen/checklist.html
\textsuperscript{22}http://www.ncbi.nlm.nih.gov/genome/annotation_euk/process/
Finally, PharmGKB contains 63 HGVS variants where the reference sequence is an arbitrary GenBank accession number. Neither BioCommons and Counsyl could infer a chromosomal position for any of them. Table 6.4 contains the results from the rest of the tools. In general we encountered several problems with this kind of variants. First of all, certain GenBank accession numbers (like M61857.1 and U56438.1) seem to cause the current version of Mutalyzer to crash. For other accession numbers (like AY313163.1) Mutalyzer informs the user that it contains more than one gene, but it does not specify which these are. Even when we declared the specific gene (e.g. AY313163.1(SCN5A):c.-1418T>C) Mutalyzer failed again. Moreover, Mutalyzer didn’t have the U14680.1 transcript in its database that encodes the widely studied BRCA1 gene. For all these cases MutationInfo’s BLAT method succeeded in inferring a chromosomal position. On the other hand, this method uses BioPython to perform a ‘c. to g.’ conversion that failed for the X17059.1 entry that encodes the NAT1 gene. It also failed for the AF280107.1 sequence that contains more than one gene. Interestingly, in cases where Mutalyzer failed MutationInfo’s BLAT method succeeded and vice versa. Moreover, although Mutalyzer succeeded in performing a ‘c. to g.’ conversion for 21 out of the total 63 variants, it failed to infer a chromosomal position for any of them, and MutationInfo used the BLAT method for this purpose. Finally, TransVar and Variation Reported performed very poorly as they only located 3 and 1 variants respectively.
<table>
<thead>
<tr>
<th>Gene</th>
<th>GenBank</th>
<th>Mutation Info’s BLAT</th>
<th>Mutalyzer + BLAT</th>
<th>TransVar</th>
<th>Variation Reporter</th>
</tr>
</thead>
<tbody>
<tr>
<td>CYP2C9</td>
<td>M61857.1</td>
<td>36/36</td>
<td>Mutalyzer crashed</td>
<td>0/36</td>
<td>0/36</td>
</tr>
<tr>
<td>NAT1</td>
<td>X17059.1</td>
<td>Could not perform c. to g.</td>
<td>2/2</td>
<td>1/2</td>
<td>0/2</td>
</tr>
<tr>
<td>CYP1B1</td>
<td>U56438.1</td>
<td>1/1</td>
<td>Mutalyzer crashed</td>
<td>1/1</td>
<td>0/1</td>
</tr>
<tr>
<td>SCN5A</td>
<td>AY313163.1</td>
<td>4/4</td>
<td>No gene specified. Please choose from: (empty list)</td>
<td>0/4</td>
<td>0/4</td>
</tr>
<tr>
<td>CYP3A4</td>
<td>AF280107.1</td>
<td>Parenthesis format</td>
<td>8/8</td>
<td>0/8</td>
<td>0/8</td>
</tr>
<tr>
<td>CYP3A7</td>
<td>AF280107.1</td>
<td>Parenthesis format</td>
<td>11/11</td>
<td>1/11</td>
<td>0/11</td>
</tr>
<tr>
<td>BRCA1</td>
<td>U14680.1</td>
<td>1/1</td>
<td>Accession Number could not be found</td>
<td>0/1</td>
<td>1/1</td>
</tr>
</tbody>
</table>

Table 6.4: Results from attempting to locate chromosomal positions for HGVS variants with GenBank reference accession numbers. The table includes tools that succeeded for at least for one variant. The table reports either the number of successful vs. total variants identified (for example 1/1), or the reason why they failed (i.e. crashed).

### 6.4.2.1 Comparison between existing tools and MutationInfo’s pipeline

The PharmGKB database contains 634 HGVS variants of interest in pharmacogenomics. Assuming that researchers have access to and knowledge of all the tools presented in this paper, they could directly infer chromosomal positions for 502 of these, i.e. 79%. This percentage also assumes that the researchers identify and correct the parsing
problems that we located for 206 variants (see Table 6.1). With MutationInfo’s pipeline they could directly get the positions for 628 (99%) of these variants without performing any quality check or preprocessing. The 6 variants that fail MutationInfo’s pipeline are erroneous entries, 3 of them can be easily corrected.

The difference between MutationInfo and other tools stems mainly from the fact that existing tools focus only on specific types of HGVS variants. Mutalyzer, BioCommons and Counsyl perform very well when HGVS variants have reference sequences with NM_ accession numbers. Yet, this is the case for only 32% of the variants present in PharmGKB. As we described, the consortia responsible for cataloging and curating the variants and haplotypes for genes with known pharmacogenetics implications list NT_, NG_, NC_ and arbitrary GenBank accession numbers as their reference sequences (covering in total 68% of variants in PharmGKB). If we omit NC_ variants (9% of total) where the conversion is simple then there are 368 variants (58%) and existing tools can infer a position for only 144 of them (39%).

Among all existing tools, Mutalyzer was the most successful at parsing and inferring a genetic position (c. to g. conversion) since it performed this task for 527 variants (83%). Nevertheless, Mutalyzer failed to infer a chromosomal position for 320 of these variants and a BLAT search was required. It is interesting that in Mutalyzer’s initial report (See Table 2 in Wildeman et al. [38]), more than half of the variants failed parsing in three different databases (HbVar, BRCA2 and PAH). They also mention: “Mutalyzer can only be effective when variant descriptions in HGVS-format are combined with a well-annotated genomic reference sequence”. This underlines the necessity of a tool like MutationInfo that applies alternative methods (BLAT search) when similar tools fail.

The most successful tool that inferred chromosomal positions without the use of a BLAT search was Variation Reported, which completed this task for 408 variants (65%). Nevertheless if we use MutationInfo’s BLAT service without any other tools, we will be able to infer positions for 476 out of the 634 variants (75%).

6.5 Analysis of HGVS variants with gene names

The majority of mutations that are reported in scientific reports use gene names instead of reference sequences. Mostly researchers that focus on specific diseases, genes, pathways or biomolecules, treat gene names as univocal reference systems. Historically, this can be attributed to the fact that gene names are more recognizable than database accession entries of reference sequences. Yet this practice is explicitly discouraged by the official HGVS recommendations [10]. In their report, the second of seven most important recommendations is: “The accession number in primary
sequence databases (GenBank, EMBL, DDJB, SWISS-PROT) should be mentioned in the publication/database submission. When available, the genomic reference sequence is preferred. For each gene, a reference sequence needs to be established.

Here we report on the application of the MutationInfo pipeline on HGVS variants found in the abstracts of scientific publications. The purpose of this experiment was to assess the genome-wide translatability of reported HGVS mutations and to also measure the ability of MutationInfo to infer the chromosomal position of HGVS variants that use gene names. In order to obtain a large collection of published variants, we queried the European PubMed Central (PMC) [6] for all abstracts that contained the word “mutation” and we downloaded the resulting 1 million abstracts. Then we extracted all the HGVS mutations that represented simple nucleotide substitutions. We applied a simple, regular expression search that matched the position and alteration part of HGVS mutations that defined a substitution (for example, 76A>C, 88+1G>T, -14G>C). We then looked for coding or genomic position identifiers (c. and g. respectively, if none were found we assumed c.) in the text preceding the matching pattern and we also looked for gene names in the text surrounding this pattern. With this information at our disposal we formed valid HGVS mutation identifiers. The complete Python code for this experiment is available as a Jupyter notebook in the GitHub repository of MutationInfo. Of course, this methodology captures only a small number of the possible HGVS mutations in abstracts, since it does not catch insertions, deletions, duplications or other more complex forms of variation. Nevertheless, our purpose was not to measure the level of author compliance to HGVS recommendations, but rather to measure the accuracy of HGVS mutations that use gene names. In total, we extracted 42,114 valid HGVS mutations with gene names. Of course many of these mutations were mismatches of our pattern-matching algorithm. To filter our results, we tested all mutations on both VEP and TransVar and we removed 8,777 that did not return a valid result from either method. In total, 21,503 mutations yielded valid results (including chromosomal positions) from both tools, 10,764 were validated only by TransVar and 1,070 were validated only by VEP. As we stated above, results from TransVar include all the possible transcripts that match an HGVS position. Overall, 32% of variants had at least two transcripts, with an average number of 1.58 transcripts per variant. Also, of the 32,267 variants matched by TransVar, only 19,048 (59%) had the same reference sequence nucleotide in all transcripts as the initial HGVS mutation found in the abstracts. Additionally, 9,226 TransVar results (29%) had at least one transcript that had a reference nucleotide that was different to the one reported in the abstract. The remaining 12% of TransVar results all had discordant reference nucleotides.

Variants that were processed by both TransVar and VEP showed a remarkable concordance. Of 21,503 variants in this category, 21,278 (0.99%) were allotted the same
reference and chromosomal position by both tools. Although 3,427 variants (16%) were allotted at least one additional transcript by TransVar, with a different location from the one allotted by VEP. In Figure 6.3 we present a summary of these findings in a set of three tables.
Figure 6.3: A summary of results from the MutationInfo pipeline for abstracts that contained HGVS variants with gene names. Table A presents the concordance between VEP and TransVar for 21,503 HGVS variants that were validated by both tools. Table B presents a contingency table showing the number of variants that were validated by either VEP or TransVar. In total, 33,337 HGVS variants were validated by at least one tool. Table C presents the concordance between the reference nucleotide in the multiple transcripts found by TransVar and the nucleotide reported in the abstract.
Guidelines for correct HGVS processing

- Support for many RefSeq accession numbers (NM_, NC, NT_, NM_, NG_, ...).
- Support for arbitrary GenBank accession numbers.
- Support for Ensembl and RefSeq accession numbers.
- Support for LRG accession numbers (officially recommended by HGVS).
- Support for gene names as reference transcripts and show all possible locations.
- Support for explicit HGVS forms like chr1:123456A>C.
- Support for multiple position identifiers (c., g., r., p.).
- Support for many variant kinds (Substitutions, Indels, Inversions, ...).
- Make visible which version of HGVS guidelines the tool adheres to.
- Provide basic quality control indicators, like correctness of reference sequence.
- Offer conversion between reference sequences that include chromosome sequences.

Table 6.5: List of guidelines for correct HGVS processing. According to our experiments no tool adheres to all these guidelines.

If we define as absolutely concordant those HGVS variants that were allotted the same chromosomal position by both tools, with no multiple transcripts and a reference nucleotide equal to the one reported in the abstract, then only 54% (17,851 out of 33,337) achieved this property. Moreover, out from 42,114 valid HGVS mutations only 880 included a reference sequence (2%) from which only 652 included also the version of the sequence. Moreover, of 42,114 valid HGVS mutations, only 880 also included a reference sequence (2%) in the same abstract of which only 652 also included the version of the sequence. Interestingly, for 17 of these abstracts, the position that was returned by VEP when using the gene name was different to the position returned by Mutalyzer when using the reference sequence. This highlights the fact that gene names should be avoided for HGVS mutations. The MutationInfo pipeline reports not only the chromosomal position of these variants, but also shows any detected inconsistencies.
6.6 Conclusions

We expect the quality of reported genetic variants to improve as the process of identifying genetic variants becomes more automated using high throughput techniques like Next Generation Sequencing [31]. Issues like ambiguity of the location or the alternative sequence of a mutation will be tackled sufficiently by existing pipelines for variant discovery [12]. Yet there is still a large and valuable legacy of identified variants discovered with less automated techniques, obsolete technology or simply by following incorrect reporting guidelines. Since, the impact of variant discovery in clinical practice is increasing, the availability of software tools for performing uniform quality checks and extracting meta-information from reported variants is also becoming more important. MutationInfo is a tool that belongs in this category for the purpose of quickly recovering (or validating) the chromosomal position, the reference and alternative sequence of a variant. It is mainly a frontend for existing tools and tries to combine them for this purpose while offering a ‘low-level’ BLAT search method if all other tools fail. The application of MutationInfo in PharmGKB, which is a typical human curated LSMD of high significance, demonstrated that existing tools have substantial limitations in identifying positional information and revealing errors in database entries. Additionally, the very common practice of using gene names instead of sequence accession numbers can have a serious detrimental effect on the translatability of these variants. Put simply, researchers cannot be absolutely confident of the location of a variant of this kind in approximately half of the cases reported.

This finding alerts us to a more generic issue: reporting a variant in a database or in a scientific publication is, in fact, a very important statement that affects other people’s research and may even influence the diagnosis of a disease and the choice of therapeutic actions for patients. Since querying biomedical bibliographic databases, like PubMed, is a primary method for locating disease-related variants, researchers must take their responsibility very seriously when reporting these variants. On the other hand, researchers should not rely on the accuracy of a reported variant with full confidence, at least not for inferring its chromosomal position. Instead they should always apply quality-control and annotation protocols like these aggregated in the MutationInfo tool.

We have also compiled a list of guidelines (Table 6.5) that future tools should follow in order to successfully parse and process HGVS variants. Since, at present, no individual tool follows all the guidelines on this list, we propose here the use of a pipeline that combines most of the existing tools for this task. MutationInfo is an open source tool that tries to fill this gap. It targets bioinformaticians with minimum knowledge of the Python programming language. It contains an installation script that manages
all dependencies and offers a single function (get_info) for accessing its methods. Finally MutationInfo offers a web service that can be easily accessed by individually researchers or embedded in existing online services.

In our future work we plan to augment MutationInfo’s pipeline with a liftover method as well as with additional tools (our first priority is Variation Reporter). We also plan to make MutationInfo compatible with the latest version of HGVS nomenclature guidelines, including the extensions suggested by the authors of Mutalyzer, that it can support more complex structural changes [35].

Acknowledgments

Funding: The research leading to these results has received funding from the Ubbo Emmius Fund to AK, eMoDiA (electronic Molecular Diagnostics Assistant) project (11_SYN10145) under the ‘Competitiveness and Entrepreneurship’ (OPCE II Greek-EU) operational program to EK, GP, the RD-Connect project (FP7-305444; European Commission) to TK, GPP, BBMRI-NL, a research infrastructure financed by the Netherlands Organization for Scientific Research (NWO project 184.021.007), to MB and MS and NWO VIDI grant number 917.164.455 to MS.
Bibliography


[38] Martin Wildeman, Ernest van Ophuizen, Johan T den Dunnen, and Peter E M Taschner. Improving sequence variant descriptions in mutation databases and


