vmvt Documentation

Release 0.9.4

Peter Robinson

Apr 22, 2021

Tutorial

1 Quick Example 3

2 Feedback 5

vmvt is a modern Java (version 11 and above) library for creating SVG plots to represent splice variants. It can be used as a library or as a standalone app.

CHAPTER 1

Quick Example

The following snippet will create a splice donor Sequence Walker (for details, see *Sequence walkers*).

```
import org.monarchinitiative.vmvt.core;

VmvtGenerator vmvt = new VmvtGenerator();

// create a Splice Donor sequence logo
String logoSvg = vmvt.getDonorLogoSvg();

// create a sequence walker showing the effects of a variant
final String ref = "AAGGTCAGA";
final String alt = "AAGATCAGA";
String walkerSvg = vmvt.getDonorWalkerSvg(ref,alt);
```

CHAPTER 2

Feedback

The best place to leave feedback, ask questions, and report bugs is the vmvt Issue Tracker.

2.1 Sequence Logos

In 1990, Tom Schneider introduced Sequence logos as a way of graphically displaying consensus sequences. The characters representing the sequence are stacked on top of each other for each position in the aligned sequences. The height of each letter is made proportional to its frequency, and the letters are sorted so the most common one is on top. The height of the entire stack is then adjusted to signify the information content of the sequences at that position. From these 'sequence logos', one can determine not only the consensus sequence but also the relative frequency of bases and the information content (measured in bits) at every position in a site or sequence. The logo displays both significant residues and subtle sequence patterns (Nucleic Acids Res 1990;18:6097-100).

Fig. 1: Sequence Logo graphic for a donor variant.

2.1.1 Creating Sequence Logos with vmvt

The following code creates a splice donor Logo.

```
import org.monarchinitiative.vmvt.core;

VmvtGenerator vmvt = new VmvtGenerator();
String donor = vmvt.getDonorLogoSvg();
```

A splice acceptor logo is created as follows.

```
import org.monarchinitiative.vmvt.core;

VmvtGenerator vmvt = new VmvtGenerator();
String donor = vmvt.getAcceptorLogoSvg();
```

2.2 Sequence rulers

Sequence rulers are SVG graphics that show the sequence of the donor or acceptor site, mark the intron-exon boundary, and show the position of any alternate bases that diverge from the reference sequence.

Fig. 2: Sequence ruler graphic for a donor variant.

2.2.1 Creating Sequence Rulers with vmvt

The following code creates a splice donor ruler. Note that the input Strings must be 9 nucleotides long. Vmvt will treat the string as corresponding to positions (-3,+6) of the intron-exon boundary of a splice donor sequence. Sequences can be provided in upper or lower case.

```
import org.monarchinitiative.vmvt.core;

final String ref = "AAGGTCAGA";
final String alt = "AAGATCAGA";

VmvtGenerator vmvt = new VmvtGenerator();
String svg = vmvt.getDonorSequenceRuler(ref,alt);
```

The acceptor version is created analogously with the `getAcceptorSequenceRuler` command.

2.3 Sequence walkers

Tom Schneider introduced Sequence Walkers in 1995 as a way of graphically displaying how binding proteins and other macromolecules interact with individual bases of nucleotide sequences. Characters representing the sequence are either oriented normally and placed above a line indicating favorable contact, or upside-down and placed below the line indicating unfavorable contact. The positive or negative height of each letter shows the contribution of that base to the average sequence conservation of the binding site, as represented by a sequence logo (Nucleic Acids Res 1997;25:4408-15). In 1998, Peter Rogan introduced the application of individual information content and Sequence Walkers to splicing variants (Hum Mutat 1998;12:153-71).

Our version of the sequence walker combines the reference and the alternate sequence. The positions in which the alternate differs from the reference are indicated by a grey box and both nucleotides are shown. In many disease-associated variants, the reference base will be position upright and the alternate base will be positioned beneath the line.

Fig. 3: Sequence Walker graphic for a +1G>A donor variant.

2.3.1 Creating Sequence Walkers with vmvt

The following code creates a splice donor walker. Note that the input Strings must be 9 nucleotides long. Vmvt will treat the string as corresponding to positions (-3,+6) of the intron-exon boundary of a splice donor sequence. Sequences can be provided in upper or lower case.

```
import org.monarchinitiative.vmvt.core;

final String ref = "AAGGTCAGA";
final String alt = "AAGATCAGA";

VmvtGenerator vmvt = new VmvtGenerator();
String svg = vmvt.getDonorWalkerSvg(ref,alt);
```

A splice acceptor walker is created as follows. Input Strings must be 27 nucleotides long.

Fig. 4: Sequence Walker graphic for a +1G>A Acceptor variant.

Vmvt will treat the string as corresponding to positions (-25,+2) of the intron-exon boundary of a splice acceptor sequence. Sequences can be provided in upper or lower case.

```
import org.monarchinitiative.vmvt.core;

final String ref = "cctggctggcggcaccgggtgccagGT";
/** chr10-90768644-A-G, -2 position */
final String alt = "cctggctggcggcaccgggtgccggGT";
VmvtGenerator vmvt = new VmvtGenerator();
String svg = vmvt.getAcceptorWalkerSvg(ref,alt);
```

2.4 Sequence trekkers

We combine the sequence logo (see *Sequence Logos*) and walker (see *Sequence walkers*) in a new figure that we call sequence trekker (because a trek goes further than a walk).

Fig. 5: Sequence Trekker graphic for a +1G>A donor variant.

2.4.1 Creating Sequence Trekkers with vmvt

The following code creates a splice donor walker. Note that the input Strings must be 9 nucleotides long. Vmvt will treat the string as corresponding to positions (-3,+6) of the intron-exon boundary of a splice donor sequence. Sequences can be provided in upper or lower case.

```
import org.monarchinitiative.vmvt.core;

final String ref = "AAGGTCAGA";
final String alt = "AAGATCAGA";

VmvtGenerator vmvt = new VmvtGenerator();
String svg = vmvt.getDonorTrekkerSvg(ref,alt);
```

2.5 Sequence trekkers with Ri

This graphic can be used to illustrate the effects of a variant that simultaneously creates a cryptic splice site and also weakens the canonical splice site. Please see *Sequence trekkers* for information about the Trekker.

Consider the variant NM_000518.5(HBB):c.90C>T (p.Gly30=). This variant is listed as Pathogenic in Clinvar (VCV000038682.2), and may create a novel 5' splicing donor site and a frameshift change. The variant is located at the -3 position of the donor site of exon 1 of the hemoglobin beta (HBB) gene. The variant is related to beta Thalassemia.

VMVT models the donor site as a 9 nucleotide sequence made up of the last three bases of the upstream exon and the first 6 bases of the following intron. The above variant (shown in brackets) affects the very first base of the canonical (wildtype) donor sequence:

```
[C/T]AG|GTTGGT
```

The variant creates a novel (cryptic) splice site three bases upstream, shifting the reading frame:

```
CTG|G[C/T]AGGT
```

To represent both changes with VMVT, use the following code. The corresponding SVG graphics are shown beneath the Java code.

1. Canonical donor site

```
import org.monarchinitiative.vmvt.core;

final String ref = "CAGGTTGGT";
final String alt = "TAGGTTGGT";

VmvtGenerator vmvt = new VmvtGenerator();
String svg = vmvt.getDonorWithRi(ref, alt);
```

Fig. 6: Sequence Trekker graphic for NM_000518.5(HBB):c.90C>T.

1. Canonical donor site

```
import org.monarchinitiative.vmvt.core;

final String ref2 = "CTGGCAGGT";
final String alt2 = "CTGGTAGGT";
vmvt = new VmvtGenerator();
svg = vmvt.getDonorWithRi(ref2, alt2);
```

Fig. 7: Sequence Trekker graphic showing a cryptic splice site created by NM_000518.5(HBB):c.90C>T.

2.6 Illustrating Exonic Splicing Enhancers

Exons may contain splicing regulatory elements (ESRs), such as exonic splicing enhancers (ESEs) and exonic splicing silencers (ESSs). ESEs and ESSs usually correspond to 6–8 nucleotide stretches that serve as binding sites for splicing activator or splicing repressor proteins. Variants that occur in ESEs or ESSs can cause splicing defects such as exon skipping (PLoS Genet 2016;12:e1005756).

2.6.1 Hexamer analysis

In 2011, Shengdong Ke and colleagues investigated a complete set of RNA 6-mer sequences (hexamers) by deep sequencing successfully spliced transcripts. All 4096 6-mers were substituted at five positions within two different

internal exons in a 3-exon minigene, and millions of successfully spliced transcripts were sequenced after transfection of human cells and the splicing effect of each 6-mer could be quantified (Genome Res 2011;21:1360-74).

Daniela Di Giacomo and colleagues leveraged these hexamer scores to investigate candidate disease-associated variants by summing the the exonic splicing regulator sequence scores (ESRseq Scores) of the hexamers that overlap the variant position for the the reference sequence (ESRseq WT) and the alternate sequence (ESRseq VAR) and then calculating the change in total ESRseq score (i.e., ESRseq VAR minus ESRseq WT). The authors showed that experimental data had high sensitivity in detecting variants that increased exon skipping (Hum Mutat 2013;34:1547-57), a finding that was confirm by Omar Soukarieh and colleagues, who terms this score ΔtESRseq (PLoS Genet 2016;12:e1005756). For simplicity, we will refer to this score as ΔESR.

In the following example, we show a synonymous variant in exon 39 of FBN1 (NM_000138.4:c.4773A>C) that was shown to causes Exon Skipping and to be associated with Marfan syndrome (Li et al., 2020). The variant is associated with a ΔtESR of 0.22. The following graphic shows how the score was calculated from the ESRseq of the six hexamers that overlap with the variant.

Fig. 8: Hexamer graphic for the variant NM_000138.4:c.4773A>C.

2.6.2 Creating Hexamer Plots with vmvt

The following code creates a hexamer plot. Note that the input Strings must be 11 nucleotides long for SNVs, whith the variant being at the middle position. Sequences can be provided in upper or lower case.

```
import org.monarchinitiative.vmvt.core;

String ref = "CCTGGAGGGGA";
String alt = "CCTGGCGGGGA";
VmvtGenerator vmvt = new VmvtGenerator();
String svg = vmvt.getHexamerSvg(ref, alt);
```

2.6.3 Heptamer analysis

In 2018, Shengdong Ke performed an analysis of 7-mer sequences (heptamers), whereby each 7-mer was assigned a normalized z-score as a measure of affinity relative to the mean of all 7-mers (Genome Res 2018;28:11-24). The following figure shows an analogous analysis of NM_000138.4:c.4773A>C.

Fig. 9: Heptamer graphic for the variant NM_000138.4:c.4773A>C.

2.7 Delta Ri Score

The individual sequence information of a sequence and an alternate sequence are showing using the Sequence Walker (see *Sequence walkers*). This graphic shows the value of the difference between the reference sequence and an alternate sequence as well as the distribution of random changes to sequences of the same length. A variant that reduces the sequence information is associated with a positive delta-Ri score.

Fig. 10: Delta graphic for a +1 donor variant.

2.7.1 Creating Delta Plots with vmvt

The following code creates a delta plot. Note that the input Strings must be 9 nucleotides long. Vmvt will treat the string as corresponding to positions (-3,+6) of the intron-exon boundary of a splice donor sequence. Sequences can be provided in upper or lower case.

```
import org.monarchinitiative.vmvt.core;

final String ref = "AAGGTCAGA";
final String alt = "AAGATCAGA";

VmvtGenerator vmvt = new VmvtGenerator();
String svg = vmvt.getDelta(ref,alt);
```

The command works for both donor and acceptor sequences (the code figures out if the sequences passed to it are donor sequences (i.e., 9 nt) or acceptor sequences (i.e., 27 nt).

2.8 vmvt command line interface

vmvt has a core module intended to be used as a programming library as well as a command line interface (in the vmvt-cli module). vmvt currently has five commands, which can be seen using the -h flag.

```
$ java -jar vmvt-cli/target/vmvt-cli.jar  -h
Usage: vmvt [-hV] [COMMAND]
    Variant-motif visualization tool.
    -h, --help      Show this help message and exit.
    -V, --version   Print version information and exit.
Commands:
    logo, L    Create sequence logo
    ruler, R   Create sequence ruler
    walker, W  Create sequence ruler
    ese, E     Create ESE svg
    delta, D   Create Delta svg
```

Each of the commands has its own help menu, e.g.,

```
$ java -jar vmvt-cli/target/vmvt-cli.jar  logo -h
Usage: vmvt logo [-adhV] [-o=<outname>]
Create sequence logo
    -a, --acceptor
    -d, --donor
    -h, --help            Show this help message and exit.
    -o, --out=<outname>
    -V, --version         Print version information and exit.
```

2.9 Contributing

Contributions are welcome!

You can contribute in many ways:

2.9.1 Types of Contributions

Report Bugs

Report bugs at https://github.com/TheJacksonLaboratory/vmvt/issues

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the Github issues for bugs. If you want to start working on a bug then please write short message on the issue tracker to prevent duplicate work.

Implement Features

Look through the Github issues for features. If you want to start working on an issue then please write short message on the issue tracker to prevent duplicate work.

Write Documentation

Phenol could always use more documentation, whether as part of the official vcfpy docs, in docstrings, or even on the web in blog posts, articles, and such.

Phenol uses Sphinx for the user manual (that you are currently reading). See *doc_guidelines* on how the documentation reStructuredText is used. See *doc_setup* on creating a local setup for building the documentation.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/TheJacksonLaboratory/vmvt/issues

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

2.9.2 Documentation Guidelines

For the documentation, please adhere to the following guidelines:

- Put each sentence on its own line, this makes tracking changes through Git SCM easier.
- Provide hyperlink targets, at least for the first two section levels.
- Use the section structure from below.

```
.. heading_1:

=========
Heading 1
=========


.. heading_2:

---------
Heading 2
---------


.. heading_3:

Heading 3
=========


.. heading_4:

Heading 4
---------


.. heading_5:

Heading 5
~~~~~~~~~


.. heading_6:

Heading 6
:::::::::
```

2.9.3 Documentation Setup

For building the documentation, you have to install the Python program Sphinx. This is best done in a virtual environment. The following assumes you have a working Python 3 setup.

Use the following steps for installing Sphinx and the dependencies for building the phenol documentation:

```
$ cd phenol/manual
$ virtualenv -p python3 .venv
$ source .venv/bin/activate
$ pip install sphinx sphinx_rtd_theme
```

Use the following for building the documentation. The first two lines are only required for loading the virtualenv. Afterwards, you can always use `make html` for building.

```
$ cd phenol/manual
$ source .venv/bin/activate
$ make html  # rebuild for changed files only
$ make clean && make html  # force rebuild
```

2.9.4 Get Started!

Ready to contribute? First, create your Java/Documentation development setup as described in *install_from_source/doc_setup*.

1. Fork the *vmvt* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/vmvt.git
```

3. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making your changes, make sure that the build runs through. For Java:

   $ mvn package

   For documentation:

   $ make clean && make html

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

2.9.5 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated.
3. Describe your changes in the `CHANGELOG.rst` file.

2.10 Authors

- Peter N. Robinson
- Daniel Danis

2.11 Changelog

2.11.1 v0.9.4

- Draw the most common bases on the top of the character stack in the sequence logo

2.11.2 v0.9.3

- Add IC bar charts
- Sequence rulers with custom offsets
- Add Maven wrapper, fix javadoc plugin configuration

2.11.3 v0.9.1

- Fixed bug in Delta SVG when R_i is larger than the maximum SNV change

2.11.4 v0.9.0

- New SVG graphic for cryptic sites overlapping with canonical splice sites
- Jaspar parser

2.11.5 v0.8.5

- Added sequence delta output documentation and cli.
- Added graphic for showing cryptic/canonical sites with Ri

2.11.6 v0.8.0

- improved functions for display of upper level HPO categories

2.11.7 v0.6.0

- core and cli modules
- adding picocli support

2.12 License

vmvt is licensed under the BSD Clear 3-Clause License.

```
BSD 3-Clause License

Copyright (c) 2020, The Jackson Laboratory
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this
   list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice,
   this list of conditions and the following disclaimer in the documentation
```

(continues on next page)

(continued from previous page)

```
   and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its
   contributors may be used to endorse or promote products derived from
   this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

2.13 Building vmvt

The vmvt GitHub repository contains two modules (core and cli). The core module is used for the actual logic, and the cli module provides a simple command line interface. Both modules can be built using maven, as usual. The following command builds the library and the app and displays a help message on the shell.

```
git clone https://github.com/TheJacksonLaboratory/vmvt
cd vmvt
java -jar vmvt-cli/target/vmvt.jar -h
```

2.13.1 Using jpackage

vmvt is a Java 11 app. We are experimenting with jpackage to build standalone command-line interface applications. For this, we need to download the jpackage app for the OS for which we desire to create the standalone app. Gluon has made these packages available for Java 11.

- linux
- Mac
- Windows

2.13.2 Creating an installer for Mac

We will demonstrate the process for the Mac. Note that we are assuming that you have Java 11 and Java 14 installed on your machine.

First, set `JAVA_HOME` (temporarily) to the value for Java 14.

```
export JAVA_HOME=`/usr/libexec/java_home -v 14`
```

You can check that the above command has worked by entering `echo $JAVA_HOME`, which should reveal the correct path (e.g., `/Library/Java/JavaVirtualMachines/jdk-14.0.1.jdk/Contents/Home`).

We can now run jpackage as follows (adjust the path to the downloaded jpackage executable as needed). First change directory into the `vmvt-cli` subdirectory and then execute this command.

```
/Library/Java/JavaVirtualMachines/jdk-14.0.1.jdk/Contents/Home/bin/jpackage \
    --input target/ \
    --name VmvtCli \
    --main-jar target/vmvt-cli.jar \
    --main-class org.monarchinitiative.vmvt.cli.Main \
    --type dmg
```

<Note: to do not working yet!>