mitoviz Documentation

Release 0.9.0

Roberto Preste

Jun 11, 2020

CONTENTS:

Plot variants on the human mitochondrial genome.

- Free software: MIT license
- Documentation: https://mitoviz.readthedocs.io
- GitHub repo: https://github.com/robertopreste/mitoviz

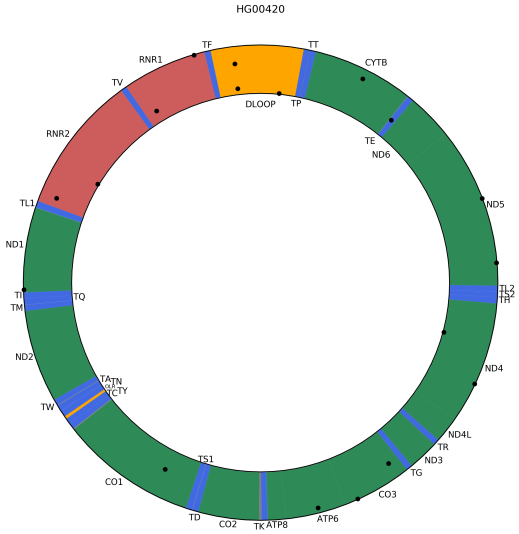
CHAPTER

ONE

FEATURES

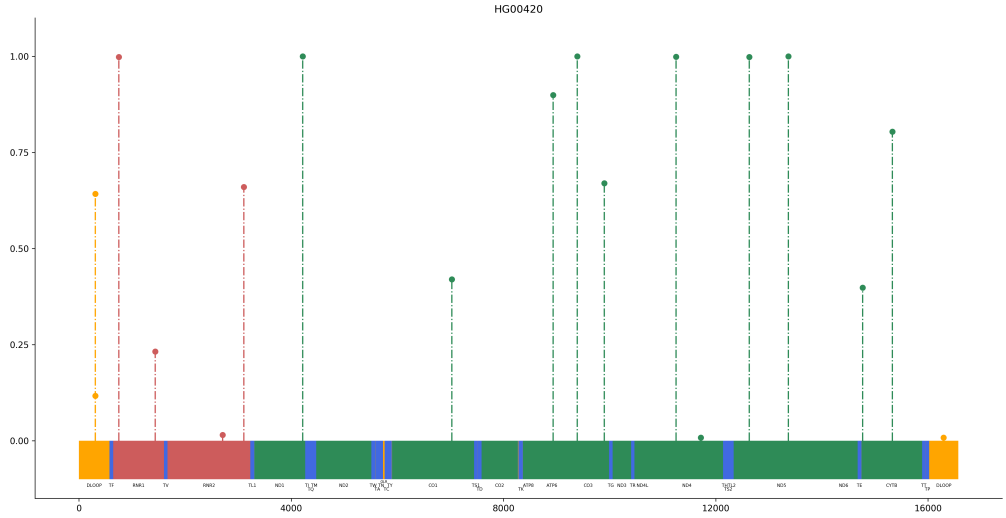
mitoviz is a simple python package to plot human mitochondrial variants on a graphical representation of the human mitochondrial genome. It currently supports plotting variants stored in VCF and tabular files, as well as from general `pandas` dataframes when importing mitoviz in Python.

Variants are shown according to their heteroplasmic fraction (HF), plotting variants with HF = 1.0 on the outer border of the mitochondrial circle, those with HF = 0.0 on the inner border and all the others in between, according to their actual HF value.



If the HF information is not available, variants will all be shown in the middle of the mitochondrial circle.

A linear representation of the mitochondrial genome can also be plotted; in this case, variants are shown using a *lollipop plot* style, with the height of the marker reflecting their HF.



Variants with no HF information will be shown as if their HF was 0.5.

CHAPTER

TWO

USAGE

mitoviz can be used both from the command line and as a python module.

2.1 Command Line

Given a VCF file with human mitochondrial variants (`sample.vcf`), plotting them is fairly simple:

```
$ mitoviz sample.vcf
```

An image named `mitoviz.png` will be created in the current directory; if you want to provide a specific filename where the plot will be saved, just add the `--output` option with the desired path:

```
$ mitoviz sample.vcf --output my_mt_plot.png
```

Linear plots can be created using the `--linear` option:

```
$ mitoviz sample.vcf --linear
```

Polar and linear interactive plots can also be created by adding the `--interactive` option, and will be saved to an HTML file:

```
$ mitoviz sample.vcf --interactive
```

It is also possible to plot variants stored in a tabular file, such as CSV or TSV formats; mitoviz will automatically recognise them, treating the file as comma-separated by default. If a different separator is used (as in the case of TSV files), just specify it with the `--sep` option:

```
$ mitoviz sample.tsv --sep "\t"
```

If you just need to create an empty mitochondrial plot, we've got you covered: use the `mitoviz-base` command and provide one or more options like `--linear`, `--interactive`, `--legend`, `--split`, `--output`, based on your needs.

2.2 Python Module

Import mitoviz and use its `plot_vcf` function to use it in your own script:

```
from mitoviz import plot_vcf

my_plot = plot_vcf("sample.vcf")
```

In this case, no plot will be shown until a call to `plt.show()` is made. It is possible to save the resulting plot using the `save` option and to provide a specific file where the plot will be saved using the `output` option:

```
plot_vcf("sample.vcf", save=True, output="my_mt_plot.png")
```

By default, a polar plot is returned; linear plots are easily created using the `linear` option:

```
plot_vcf("sample.vcf", save=True, linear=True)
```

Interactive plots can be created with the `interactive` option, and can be either saved to an HTML file or inspected in a Jupyter notebook:

```
# Show the interactive plot (works in a Jupyter notebook)
plot_vcf("sample.vcf", interactive=True)
# Save the interactive plot to an HTML file
plot_vcf("sample.vcf", interactive=True, save=True)
```

A similar function to plot variants contained in a pandas DataFrame is available as `plot_df`. Supposing you have a pandas DataFrame with human mitochondrial variants named `variants_df`, it is possible to plot them as follows:

```
from mitoviz import plot_df

plot_df(variants_df)
```

Variants stored in tabular files can be plotted using `plot_table`, which accepts the same options available for `plot_vcf` and `plot_df`, with the addition of `sep`, which is used to specify the column separator. By default, the comma is used as column delimiter:

```
from mitoviz import plot_table

# plotting a CSV file
plot_table("sample.csv")
# plotting a TSV (tab-separated) file
plot_table("sample.tsv", sep="\t")
```

`plot_table` also accept additional keyword options, which will be passed to `pandas.read_table` when processing the given input file:

```
plot_table("sample.tsv", sep="\t", comment="#", skiprows=0)
```

If you just need to create an empty mitochondrial plot, the `plot_base` function allows to do so, and accepts the `linear`, `interactive`, `legend`, `split`, `output` and `save` arguments to further tweak its behaviour.

Please refer to the Usage section of the documentation for further information.

CHAPTER

THREE

INSTALLATION

PLEASE NOTE: HmtNote only supports Python >= 3.6!

The preferred installation method for mitoviz is using `pip`:

```
$ pip install mitoviz
```

Please refer to the Installation section of the documentation for further information.

CHAPTER

FOUR

CREDITS

This package was created with Cookiecutter and the cc-pypackage project template.

CHAPTER

FIVE

TABLE OF CONTENTS

5.1 mitoviz

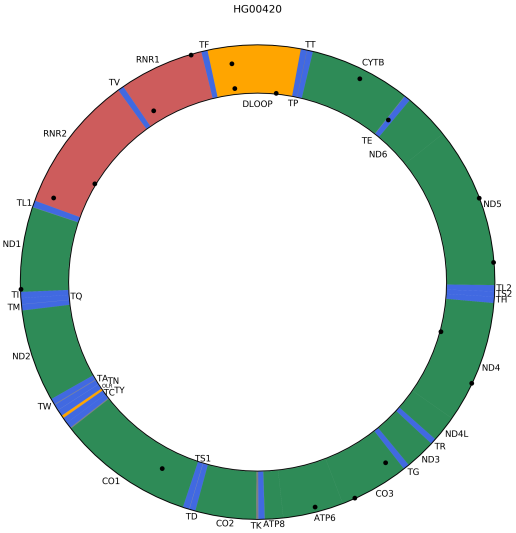
Plot variants on the human mitochondrial genome.

- Free software: MIT license
- Documentation: https://mitoviz.readthedocs.io
- GitHub repo: https://github.com/robertopreste/mitoviz

5.1.1 Features

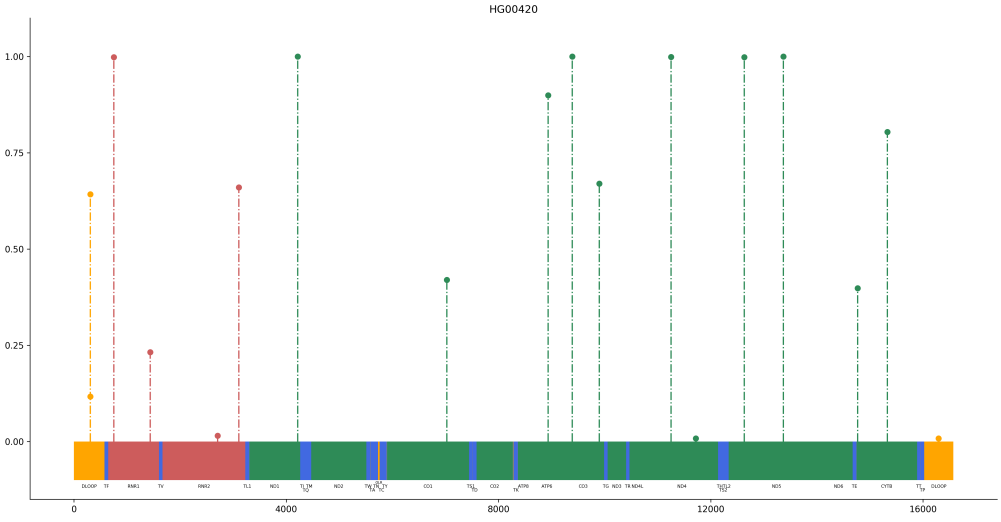
mitoviz is a simple python package to plot human mitochondrial variants on a graphical representation of the human mitochondrial genome. It currently supports plotting variants stored in VCF and tabular files, as well as from general `pandas` dataframes when importing mitoviz in Python.

Variants are shown according to their heteroplasmic fraction (HF), plotting variants with HF = 1.0 on the outer border of the mitochondrial circle, those with HF = 0.0 on the inner border and all the others in between, according to their actual HF value.



If the HF information is not available, variants will all be shown in the middle of the mitochondrial circle.

A linear representation of the mitochondrial genome can also be plotted; in this case, variants are shown using a *lollipop plot* style, with the height of the marker reflecting their HF.



Variants with no HF information will be shown as if their HF was 0.5.

5.1.2 Usage

mitoviz can be used both from the command line and as a python module.

Command Line

Given a VCF file with human mitochondrial variants (`sample.vcf`), plotting them is fairly simple:

```
$ mitoviz sample.vcf
```

An image named `mitoviz.png` will be created in the current directory; if you want to provide a specific filename where the plot will be saved, just add the `--output` option with the desired path:

```
$ mitoviz sample.vcf --output my_mt_plot.png
```

Linear plots can be created using the `--linear` option:

```
$ mitoviz sample.vcf --linear
```

Polar and linear interactive plots can also be created by adding the `--interactive` option, and will be saved to an HTML file:

```
$ mitoviz sample.vcf --interactive
```

It is also possible to plot variants stored in a tabular file, such as CSV or TSV formats; mitoviz will automatically recognise them, treating the file as comma-separated by default. If a different separator is used (as in the case of TSV files), just specify it with the `--sep` option:

```
$ mitoviz sample.tsv --sep "\t"
```

If you just need to create an empty mitochondrial plot, we've got you covered: use the `mitoviz-base` command and provide one or more options like `--linear`, `--interactive`, `--legend`, `--split`, `--output`, based on your needs.

Python Module

Import mitoviz and use its `plot_vcf` function to use it in your own script:

```
from mitoviz import plot_vcf

my_plot = plot_vcf("sample.vcf")
```

In this case, no plot will be shown until a call to `plt.show()` is made. It is possible to save the resulting plot using the `save` option and to provide a specific file where the plot will be saved using the `output` option:

```
plot_vcf("sample.vcf", save=True, output="my_mt_plot.png")
```

By default, a polar plot is returned; linear plots are easily created using the `linear` option:

```
plot_vcf("sample.vcf", save=True, linear=True)
```

Interactive plots can be created with the `interactive` option, and can be either saved to an HTML file or inspected in a Jupyter notebook:

```
# Show the interactive plot (works in a Jupyter notebook)
plot_vcf("sample.vcf", interactive=True)
# Save the interactive plot to an HTML file
plot_vcf("sample.vcf", interactive=True, save=True)
```

A similar function to plot variants contained in a pandas DataFrame is available as `plot_df`. Supposing you have a pandas DataFrame with human mitochondrial variants named `variants_df`, it is possible to plot them as follows:

```
from mitoviz import plot_df

plot_df(variants_df)
```

Variants stored in tabular files can be plotted using `plot_table`, which accepts the same options available for `plot_vcf` and `plot_df`, with the addition of `sep`, which is used to specify the column separator. By default, the comma is used as column delimiter:

```
from mitoviz import plot_table

# plotting a CSV file
plot_table("sample.csv")
# plotting a TSV (tab-separated) file
plot_table("sample.tsv", sep="\t")
```

`plot_table` also accept additional keyword options, which will be passed to `pandas.read_table` when processing the given input file:

```
plot_table("sample.tsv", sep="\t", comment="#", skiprows=0)
```

If you just need to create an empty mitochondrial plot, the `plot_base` function allows to do so, and accepts the `linear`, `interactive`, `legend`, `split`, `output` and `save` arguments to further tweak its behaviour.

Please refer to the Usage section of the documentation for further information.

5.1.3 Installation

PLEASE NOTE: HmtNote only supports Python >= 3.6!

The preferred installation method for mitoviz is using `pip`:

```
$ pip install mitoviz
```

Please refer to the Installation section of the documentation for further information.

5.1.4 Credits

This package was created with Cookiecutter and the cc-pypackage project template.

5.2 Installation

5.2.1 Stable release

To install mitoviz, run this command in your terminal:

```
$ pip install mitoviz
```

This is the preferred method to install mitoviz, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

5.2.2 From sources

The sources for mitoviz can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/robertopreste/mitoviz
```

Or download the tarball:

```
$ curl  -OL https://github.com/robertopreste/mitoviz/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

5.3 Usage

mitoviz can be used both from the command line and as a python module.

5.3.1 Command Line

Given a VCF file with human mitochondrial variants (`sample.vcf`), plotting them is fairly simple:

```
$ mitoviz sample.vcf
```

An image named `mitoviz.png` will be created in the current directory.

If you want to provide a specific filename where the plot will be saved, just add the `--output` option with the desired path:

```
$ mitoviz sample.vcf --output my_mt_plot.png
```

If the provided VCF file contains more than one sample, a separate plot will be created for each of them; if you want to plot only a specific sample, use the `--sample` option:

```
$ mitoviz multisample.vcf --sample SRR1777294
```

It is possible to show labels above each variant using the `--labels` flag:

```
$ mitoviz sample.vcf --labels
```

If you want to include the HF value of variants in the labels, add the `--labels-hf` flag:

```
$ mitoviz sample.vcf --labels --labels-hf
```

Mitochondrial loci on mitoviz plots are drawn using a green color for protein-coding, blue for tRNAs, red for rRNAs, orange for regulatory (D-Loop and L-strand origin) and grey for non-coding loci. It is possible to include a legend in the resulting plot, using the `--legend` option:

```
$ mitoviz sample.vcf --legend
```

The plot can draw loci located on H and L strands on two different levels, using the `--split` option:

```
$ mitoviz sample.vcf --split
```

mitoviz can create linear plots as well, where variants are shown using a *lollipop plot* style, using the `--linear` option:

```
$ mitoviz sample.vcf --linear
```

Linear plots can be managed and customised using the `--output`, `--sample`, `--labels`, `--legend` and `--split` options.

Polar and linear interactive plots can also be created by adding the `--interactive` option, and will be saved to an HTML file:

```
$ mitoviz sample.vcf --interactive
```

It is also possible to plot variants stored in a tabular file, such as CSV or TSV formats; mitoviz will automatically recognise them, treating the file as comma-separated by default. If a different separator is used (as in the case of TSV files), just specify it with the `--sep` option:

```
$ mitoviz sample.tsv --sep "\t"
```

Additional keyword options can be specified in the format `option=value`, and will be passed to `pandas.read_table` when processing the given input file:

```
$ mitoviz sample.tsv --sep "\t" comment=#
```

If you just need to create an empty mitochondrial plot, we've got you covered: use the `mitoviz-base` command and provide one or more options like `--linear`, `--interactive`, `--legend`, `--split`, `--output`, based on your needs:

```
# Create a base polar plot
$ mitoviz-base

# Create a base linear plot and save it as "base_linear.png"
$ mitoviz-base --linear --output "base_linear.png"

# Create an interactive linear plot with split loci
$ mitoviz-base --linear --interactive --split
```

Comprehensive help about the mitoviz CLI can be found with `mitoviz --help` and `mitoviz-base --help`.

5.3.2 Python Module

Import mitoviz and use its `plot_vcf` function to use it in your own script:

```
from mitoviz import plot_vcf

my_plot = plot_vcf("sample.vcf")
```

In this case, no plot will be shown until a call to `plt.show()` is made. It is possible to save the resulting plot using the `save` option and to provide a specific file where the plot will be saved using the `output` option:

```
from mitoviz import plot_vcf

plot_vcf("sample.vcf", save=True, output="my_mt_plot.png")
```

If the provided VCF file contains more than one sample, a separate plot will be created for each of them; if you want to plot only a specific sample, use the `sample` option:

```
from mitoviz import plot_vcf

plot_vcf("multisample.vcf", save=True, sample="SRR1777294")
```

If you want to show labels for each variant plotted, add the `labels=True` option:

```
from mitoviz import plot_vcf

plot_vcf("sample.vcf", labels=True)
```

If you also want HF values in the labels, add the `labels_hf=True` option:

```
from mitoviz import plot_vcf

plot_vcf("sample.vcf", labels=True, labels_hf=True)
```

It is possible to include a legend for loci colors in the output plot, using the `legend=True` option:

```
from mitoviz import plot_vcf

plot_vcf("sample.vcf", legend=True)
```

Loci located on the H and L strands can be shown on two separate levels, using the `split=True` option:

```
from mitoviz import plot_vcf

plot_vcf("sample.vcf", split=True)
```

Linear plots can be also created (instead of the default polar plot), using the `linear=True` option:

```
from mitoviz import plot_vcf

plot_vcf("sample.vcf", linear=True)
```

The `linear=True` option can be combined with previously described options as well.

Interactive plots can be created with the `interactive` option, and can be either saved to an HTML file or inspected in a Jupyter notebook:

```
# Show the interactive plot (works in a Jupyter notebook)
plot_vcf("sample.vcf", interactive=True)
# Save the interactive plot to an HTML file
plot_vcf("sample.vcf", interactive=True, save=True)
```

Comprehensive help about the `plot_vcf` function can be found with `help(mitoviz.plot_vcf)`.

A similar function to plot variants contained in a pandas DataFrame is available as `plot_df`. Supposing you have a pandas DataFrame with human mitochondrial variants named `variants_df`, it is possible to plot them as follows:

```
from mitoviz import plot_df

plot_df(variants_df)
```

This function expects a DataFrame with at least a reference allele, position and alternate allele columns; these are respectively called "REF", "POS" and "ALT" by default, but it is possible to use custom column names:

```
from mitoviz import plot_df

plot_df(variants_df, ref_col="position", ref_col="reference", alt_col="alternate")
```

It is possible to provide optional sample and hf (heteroplasmic fraction) columns, which are called "SAMPLE" and "HF" by default but can be customised using the `sample_col` and `hf_col` options.

Apart from this, `plot_df` accepts the same set of options available for `plot_vcf`. Comprehensive help about the `plot_df` function can be found with `help(mitoviz.plot_df)`.

Variants stored in tabular files can be plotted using `plot_table`, which accepts the same options available for `plot_vcf` and `plot_df`, with the addition of `sep`, which is used to specify the column separator. By default, the comma is used as column delimiter:

```
from mitoviz import plot_table

# plotting a CSV file
plot_table("sample.csv")
# plotting a TSV (tab-separated) file
plot_table("sample.tsv", sep="\t")
```

`plot_table` also accept additional keyword options, which will be passed to `pandas.read_table` when processing the given input file:

```
from mitoviz import plot_table

plot_table("sample.tsv", sep="\t", comment="#", skiprows=0)
```

Comprehensive help about the `plot_table` function can be found with `help(mitoviz.plot_table)`.

If you just need to create an empty mitochondrial plot, the `plot_base` function allows to do so, and accepts the `linear`, `interactive`, `legend`, `split`, `output` and `save` arguments to further tweak its behaviour:

```
from mitoviz import plot_base

# Create a base polar plot
plot_base()
# Create a base linear plot and save it as "base_linear.png"
plot_base(linear=True, save=True, output="base_linear.png)
# Create an interactive linear plot with split loci
plot_base(linear=True, interactive=True, split=True)
```

5.4 API

5.4.1 Command Line Interface

5.4.2 Python Module

Plot variants from the given VCF file.

param in_vcf path of the input VCF file

param linear plot variants on a linear plot rather than a polar one [default: False]

param sample specific sample to plot (defaults to all available samples)

param save if true, the final plot will be saved to a file [default: False]

param output path of the output file where the plot will be saved

param labels if true, add a label for each variant shown [default: False]

param labels_hf if true and *labels=True*, show HF value in each variant's label [default: False]

param legend if true, add a legend for loci colors in the plot [default: False]

param split if true, plot split H and L strands [default: False]

param interactive if true, create an interactive version of the plot [default: False]

Plot variant from the given pandas DataFrame.

param in_df input pandas DataFrame

param linear plot variants on a linear plot rather than a polar one [default: False]

param sample specific sample to plot (defaults to all available samples)

param save if true, the final plot will be saved to a file [default: False]

param output path of the output file where the plot will be saved

param labels if true, add a label for each variant shown [default: False]

param labels_hf if true and *labels=True*, show HF value in each variant's label [default: False]

param legend if true, add a legend for loci colors in the plot [default: False]

param split if true, plot split H and L strands [default: False]

param interactive if true, create an interactive version of the plot [default: False]

param pos_col column name for the variant position

param ref_col column name for the variant reference allele

param alt_col column name for the variant alternate allele

param sample_col column name for the variant sample

param hf_col column name for the variant heteroplasmic fraction

Plot variants from the given tabular file.

param in_table path of the input tabular file

param sep column delimiter used [default: ','']

param linear plot variants on a linear plot rather than a polar one [default: False]

param sample specific sample to plot (defaults to all available samples)

param save if true, the final plot will be saved to a file [default: False]

param output path of the output file where the plot will be saved

param labels if true, add a label for each variant shown [default: False]

param labels_hf if true and *labels=True*, show HF value in each variant's label [default: False]

param legend if true, add a legend for loci colors in the plot [default: False]

param split if true, plot split H and L strands [default: False]

param interactive if true, create an interactive version of the plot [default: False]

param pos_col column name for the variant position

param ref_col column name for the variant reference allele

param alt_col column name for the variant alternate allele

param sample_col column name for the variant sample

param hf_col column name for the variant heteroplasmic fraction

param **kwargs additional arguments passed to pandas.read_table()

5.5 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.5.1 Types of Contributions

Report Bugs

Report bugs at https://github.com/robertopreste/mitoviz/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. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

Write Documentation

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

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/robertopreste/mitoviz/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 :)

5.5.2 Get Started!

Ready to contribute? Here's how to set up `mitoviz` for local development.

1. Fork the `mitoviz` repo on GitHub.
2. Clone your fork locally:

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

3. Set up a virtualenv for local development:

```
$ cd mitoviz/
$ python -m venv venv
$ source venv/bin/activate
$ python -m ci install-reqs
$ python -m ci install
```

4. 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 changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox. In order to test mitoviz, you'll also need to clone the `mitoviz_testimgs` repo to the right location:

```
$ git clone https://github.com/robertopreste/mitoviz_testimgs.git mitoviz/tests/
↪imgs
$ python -m ci flake8
$ pytest
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

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.

5.5.3 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. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.5 and 3.6. Check https://travis-ci.org/robertopreste/mitoviz/pull_requests and make sure that the tests pass for all supported Python versions.

5.5.4 Tips

To run a subset of tests:

```
$ pytest tests.test_mitoviz
```

5.5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bump2version patch  # possible: major / minor / patch
$ git push
$ git push --tags
```

5.6 Credits

5.6.1 Development Lead

- Roberto Preste <robertopreste@gmail.com>

5.6.2 Contributors

None yet. Why not be the first?

5.7 History

5.7.1 0.1.0 (2019-12-27)

- First release.

5.7.2 0.2.0 (2019-12-29)

- Add functionality to plot multiple samples.

0.2.1 (2020-01-06)

- Add legend to plots and update colors.

0.2.2 (2020-01-08)

- Add option to plot variant labels.

0.2.3 (2020-01-11)

- Make legend plotting optional.

5.7.3 0.3.0 (2020-01-15)

- Add `plot_df` function to plot variants from a pandas DataFrame.

5.7.4 0.4.0 (2020-01-26)

- Add `plot_table` function to plot variants from tabular files;
- add CLI functionality to plot variants from tabular files;
- refactor code.

0.4.1 (2020-02-13)

- Refactor to use abstract classes;
- Rename internal classes to _PolarLocus and _PolarVariant.

0.4.2 (2020-02-14)

- Fix bug with non coding loci not being shown in plots.

5.7.5 0.5.0 (2020-02-19)

- Add `split` option to plot split strands on polar plots.

5.7.6 0.6.0 (2020-02-29)

- `_PolarVariant` is deprecated and replaced by `_Variant`;
- Add `linear` option to create linear plots.

0.6.1 (2020-03-02)

- Refactor and clean code;
- Add CI module for internal management.

0.6.2 (2020-03-03)

- Fix borders on linear plots.

0.6.3 (2020-03-04)

- Fix stemlines on split linear plots.

0.6.4 (2020-03-10)

- Fix loci label positions on polar plots.

5.7.7 0.7.0 (2020-03-15)

- Add `--interactive` option to create interactive plots using plot.ly;
- Implement interactive basic polar plots;
- Implement interactive split polar plots.

0.7.1 (2020-03-28)

- Implement interactive basic linear plots;
- Implement interactive split linear plots.

5.7.8 0.8.0 (2020-04-05)

- Add `mitoviz-base` command to create base mitochondrial plots.

0.8.1 (2020-04-12)

- Move test images to separate subrepo.

0.8.2 (2020-04-20)

- Add `--labels-hf` (`labels-hf=True` in Python) options to include the HF value in plot labels.

5.7.9 0.9.0 (2020-06-11)

- Refactor code to create plotting classes.

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m

INDEX

M