# SSR-viz - a toolbox to detect and visualize protein subfamily specific residues

Paul Zierep

September 17, 2018

**Abstract**

Homologous proteins can be classified into protein families. The members of a family share a similar structure and sequence. Despite their inherent similarity, individual members of the same family can adopt very specific functions, leading to a further division into subfamilies. These functional differences can often be assigned to specific residues. This information can be used for various applications, such as function-based subfamily classification, rational site-directed mutagenesis and general elucidation of mechanisms of action. Here we introduce a novel user-friendly open-source software which allows for the identification and visualization of these residues.

# Contents

# 1 Installation

## 1.1 Using Pip

SSR-viz in implemented as a standalone GUI framework. It is entirely written in Python 3. The program was successfully installed on Ubuntu 14.04/16.04/18.04 and Windows 10/8. It can be installed via PIP - the official python repository. To install SSR-viz open a terminal an type:

```
pip3 install ssrviz
```

Some packages, especially **wxPython** which are normally automatically installed by PIP can make problems, as they are using system dependencies, which can lead to errors when missing. Good advice to install wxPython can be found at: `https://wxpython.org/pages/downloads/index.html`. They also supply custom builds which work on Ubuntu 16.04 and various other Linux systems.

Nevertheless wxPython needs some system dependencies on Linux: They can be easily installed for Ubuntu 16.04:

```
sudo apt-get install libwebkitgtk-dev libgtk2.0-dev libsdl1.2-dev \
freeglut3 freeglut3-dev libnotify-dev libgstreamerd-3-dev \
libgl1-mesa-glx libglu1-mesa libgl1-mesa-dev libglu1-mesa-dev \
libgconf2-dev libsdl1.2-dev zlib1g-dev libjpeg62-dev
```

In some cases the **matplotlib** might also need some Help with **tkinter**.

```
sudo apt-get install python3-tk
```

Installation via Pip for Windows should work without additional installations.

## 1.2 Clone from GitHub

It can also be cloned from GitHub `https://github.com/PhaBiFreiburg/SSR-viz`. But then the dependencies need to be installed manually, see:
SSR-viz/documenation/requirements.txt

## 1.3 Standalone executables (easy but big)

Additionally we implemented standalone executables for Windows (tested on windows 10) and Linux (tested on Ubunt 16.04). These a much bigger then the pure python module, but ship everything needed out of the Box.

## 1.4 Mafft

The only external tool needed is mafft, an excellent alignment tool which is required to map protein structure indices to the alignment. SSR-viz runs without mafft, but for the **Add_pdb** tool the mafft executable needs to be assigned (see section 5). Dont't worry mafft is easy to install on all systems:

`https://mafft.cbrc.jp/alignment/software/`

# 2 Getting started

The SSR-viz algorithm is based on a multiple sequence alignment (MSA) file in FASTA format, which can be generated with various tools, such as Clustalo and Mafft or with a Webserver such as Mustguseal or PROMALS3D.

The topic of sequence alignment is beyond the scope of this manual. Nevertheless one should keep in mind that the quality of the alignment is crucial for the detection algorithm. (Is is difficult to interpret the importance of a position, which has more gaps then amino acids.)

Please provide the alignment in FASTA format, most tools allow this as format as an output option.

The first step is the classification of the sequences into subfamilies. This is undoubtedly the most difficult part, as it often requires to identify the specific functionality based on scientific literature or even undertake experimental validation. Dedicated databases such as UniProt and PANTHER can help to identify detailed protein functionality.

Even though there are various tools available that can cluster protein sequences, these clustering methods always apply some kind of similarity scoring, which leads in most cases to a clustering based on evolutionary relationship rather then functional similarity. This is demonstrated on an example in section 6.

Ones you collected the class information of your sequences you can add them to your alignment. The **CSV_Builder** tool allows to creates a comma separated value (CSV) file which can be used to add the class label to the sequences (see section 3).

An alignment and the CSV file is everything needed to detect subfamily specific residues in the sequences. The **SSR_plot** tool handles the actual execution of the detection algorithm, the output can be a mathplotlib style plot (see section 4) as pdf, a Javlview annotation file (which can show the results together with the alignment) as well as a 'stats.csv' file which summarizes the SSRs.

cite

In most cases it is desired to observe the SSRs in a structural context. Therefore, we also developed a tool **Add_pdb** which allows to map the indices of a protein structure file (*.pdb) to the indices of the alignment in the 'stats.csv' file (see section 5).

An overview chart which explains the setup of the three tools is shown in the poster ().

add link

# 3 CSV builder

The **CSV_Builder** handles the input and takes care, that the alignment and CSV class label file have the right formating.

The mapping scheme is shown in Fig. .

fig

## 3.1 Arguments

`Input sequence alignment file`

The alignment file with the sequences of the family. The deisred format is in FASTA fromat (clustalo), see section 6 for an example.

`Inplace FASTA conversion / Temporary alignment file name`

The **CSV_Builder** routine will remove duplicates from the alignment, as multiple identical sequences will overestimate the importance of this subfamily. The alignment can be converted inplace, meaning the original alignment is overwritten or a new alignment can be created.

`Regex extraction of the class label`

The normal **CSV_Builder** routine will create a CSV file, with an empty column for the class labels. Which must be manually filled. In some cases the class label is part of the sequence names, this labels can be extracted using regular expressions (regex) patterns. The entire scope of regex is to big for this manual, but the set of examples in the appendix 7 should help to get started. There are various tools available which can be used to test regex before usage, most text editors support regex as a search option. You can for example load the alignment file into sublime or notepad, then search with regex and if the pattern is correct, it should only highlight the desired class label. An example is shown in appendix 7.

`Output`

The name of the CSV file, by default it will be created in the same folder as the sequence file.

`Delete`

Allows to overwrite existing CSV files.

# 4 SSR plot

The