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

Paul Zierep

November 21, 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 What is SSR-viz ?

SSR-viz helps to detect residues (SSRs) which distinguish protein subfamilies from each other. SSR-viz needs a Multiple Sequence Alignment (MSA) of a protein family and a comma separated value (CSV)-file which defines the subfamilies in the alignment. For each position in the alignment a score is computed which aims to highlight positions which are conserved in a subfamily but differ significantly to other subfamilies. Various outputs can be created which help to further analyze the detected SSRs.

2 Installation

2.1 Standalone executables

We implemented standalone executables for Windows (tested on windows 10) and Linux (tested on Ubuntu 16.04). These are bigger than the pure python module, but ship everything needed out of the box.

`https://github.com/PhaBiFreiburg/SSR-viz/tree/master/standalone/`

- Windows: exe.win-amd64-3.5.zip
- Linux: exe.linux-x86_64-3.5.tar.gz

2.2 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 8/10. It can be installed via pip - the official python repository. To install SSR-viz open a terminal and type:

```
pip3 install ssrviz
```

Some packages, especially **wxPython** which are normally automatically installed by PIP apply system dependencies, which can lead to errors when missing. Good guidelines 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.

System dependencies on Linux for wxPython can be installed with the following commands (tested on Ubuntu 16.04 and 18.04):

```
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
```

Installation via pip for Windows should work without additional installations (tested on Windows 10).

Given, that installation via pip works for Windows and Linux, chances are good that SSR-viz can also be installed on a Mac, but was not tested so far.

2.3 Clone from GitHub

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

2.4 Mafft

The only external tool needed is Mafft [**katoh_MAFFT:_2002**], an 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 10.1 for further help). Mafft is easy to install on all systems:
`https://MAFFT.cbrc.jp/alignment/software/`

3 Getting started

In Section 9 a detailed use case is shown, which demonstrates the usage of each tool as well as the input and output formats. The use case can be reproduced with the provided files in `https://github.com/PhaBiFreiburg/SSR-viz/use_case`.
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 [1] and Mafft or with a Webserver such as Mustguseal or PROMALS3D. However, one should keep in mind that the quality of the alignment is crucial for the detection algorithm. Please provide the alignment in FASTA format, most tools allow this 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 function based on scientific literature or even undertake experimental validation. Dedicated databases such as UniProt and PANTHER can help to identify detailed protein function. 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 than functional similarity. This is demonstrated on an example in Section 9.

Ones you collected the class information of your sequences you can add them to your alignment. The **CSV_Builder** tool allows to creates a CSV-file which can be used to add the subfamily class label to the sequences (see Section 5).

An alignment and the CSV-file is 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 [2] style plot (see Section 7) as pdf, a Javlview annotation file [3] (which can show the results together with the alignment) as well as a 'stats.csv' file which summarizes the SSRs.

In many cases it is desired to analyze 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 8).

An overview chart which explains the setup of the three tools is shown in the Figure 1.

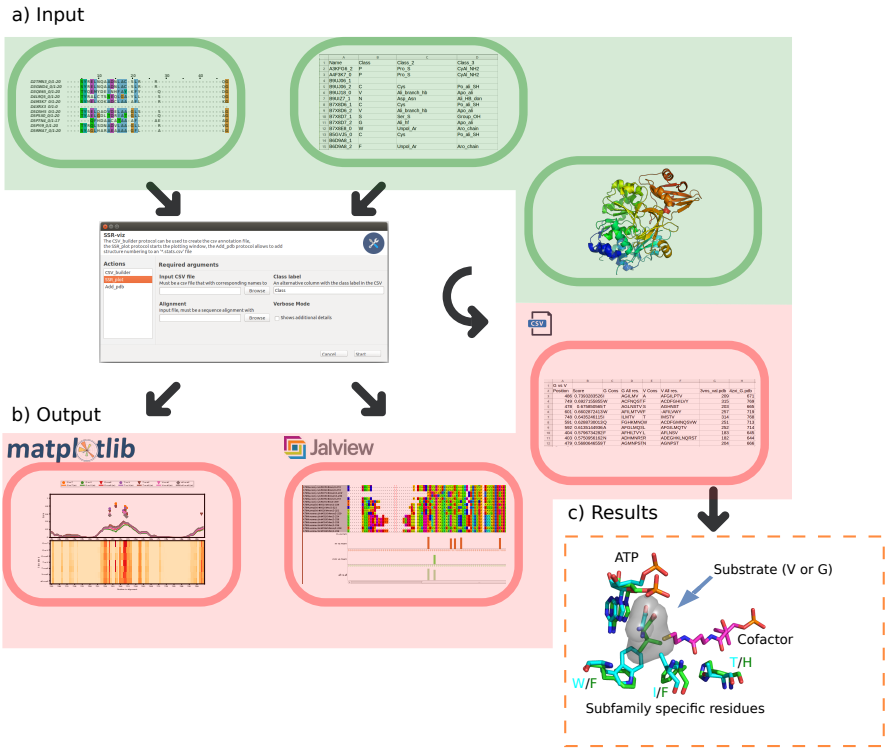


Figure 1: General flow chart of the SSR-viz toolbox. a) The input consists of a MSA together with a CSV-file holding the corresponding subfamily class labels. Multiple protein structures can also be included in order to assign the structure indices. b) The output can be created as a mathplotlib PDF, a Jalview annotation file or a summary CSV-File. This file is used to assign the indices of the structure. c) Example assignment of SSRs for the NRPS adenylation domain with specificity for Valine and Glycine.

4 Algorithm

The alignment is split into groups representing each protein subfamily as defined in the subfamily class label CSV-file. For each group a position probability matrix (PPM) is generated (see Eq. 1). This represents the alignment in form of the probability to find each amino acid at a certain position. In the following equations α and β represent PPMs for two different subfamilies.

$$PPM(\alpha) = \begin{array}{c} \\ A \\ R \\ n \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & m \end{array} \\ \begin{pmatrix} \alpha_{A1} & \alpha_{A2} & \alpha_{Am} \\ \alpha_{R1} & \alpha_{R2} & \alpha_{Rm} \\ \alpha_{n1} & \alpha_{n2} & \alpha_{nm} \end{pmatrix} \end{array} \tag{1}$$

The conservation score (CS) for each group is computed by applying the normalized Shannon entropy for each subfamily (Eq. 2). The score is inverted so that 1 represents a highly conserved position and 0 an unconserved position (Eq. 3). The mean of both subfamily represents the final CS (Eq. 4).

$$S(\alpha)_m = -\frac{\sum(\alpha_n * \log_2(\alpha_n))}{S_{max}} \tag{2}$$

$$CS(\alpha)_m = 1 - S(\alpha) \tag{3}$$

$$CS(\alpha, \beta)_m = \frac{CS(\alpha)_m + CS(\beta)_m}{2} \tag{4}$$

The difference of both subfamilies is represented by an exchange probability matrix of both PPMs (Eq. 5). This represents the probability of each residue to be exchanged with another residue in the other subfamily. The sum of the matrix results in the exchange probability (EP) score.

$$EP(\alpha, \beta)_m = \sum\sum \begin{bmatrix} 0 & \alpha_{A1} * \beta_{R1} & \alpha_{A1} * \beta_{n1} \\ \alpha_{R1} * \beta_{A1} & 0 & \alpha_{R1} * \beta_{n1} \\ \alpha_{n1} * \beta_{A1} & \alpha_{n1} * \beta_{R1} & 0 \end{bmatrix} \tag{5}$$

In order to highlight residues which are exchanged with residues with different physico-chemical properties, an additionally score is introduced. Therefore, a weighed exchange matrix (WEM) is computed based on substitution matrices such as PAM and BLOSUM [4] (Eq. 6). The matrix can be chosen as a parameter in the algorithm options.

Based on this matrix the weighed exchange probability (WEP) is computed by multiplying the EP with the matrix, thereby weighing the exchange differently based on the factor of the WEM (Eq. 7). This score is additional weighed with a factor γ in order to allow users to define the strength of the WEM individually. For some experiments a stronger focus on residues with physico-chemical exchange might be desired.

$$WEM = \begin{array}{c} \\ A \\ R \\ n \end{array} \begin{array}{c} \begin{array}{ccc} A & R & n \end{array} \\ \begin{pmatrix} 0 & i_{AR} & i_{An} \\ i_{RA} & 0 & i_{An} \\ i_{nA} & i_{nR} & 0 \end{pmatrix} \end{array} \tag{6}$$

$$WEP(\alpha,\beta)_m = \sum\sum(EP(\alpha,\beta)_m * WEM) * \gamma \tag{7}$$

The final subfamily specific residue (SSR) score of position m (ES_m) is computed by summation of all three individual scores.

$$SSR_{score}(\alpha,\beta)_m = CS(\alpha,\beta)_m * EP(\alpha,\beta)_m * WEP(\alpha,\beta)_m \tag{8}$$

The basic score represents the one-vs-one scoring scheme for two subfamily classes. For the other schemes (one-vs-all) and (all-vs-all) the average of the scores are taken. For example for a subfamily α, given three subfamilies α, β and γ the one-vs-all score for α would be computed as shown in Eq. 9.

$$\text{(one-vs-all): } SSR_{score}(\alpha)_m = \frac{SSR_{score}(\alpha,\beta)_m + SSR_{score}(\alpha,\gamma)_m}{2} \tag{9}$$

An example of the scoring algorithm for a hypothetical alignment is shown in Figure 2.

5 CSV builder

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

5.1 Arguments

`Input sequence alignment file`

The alignment file with the sequences of the family. The desired format is in FASTA format (clustalo), see Section 9 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, that means 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 labels are part of the sequence names, the labels can be extracted using regular expressions (regex) patterns. Examples in the appendix 10 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 10.

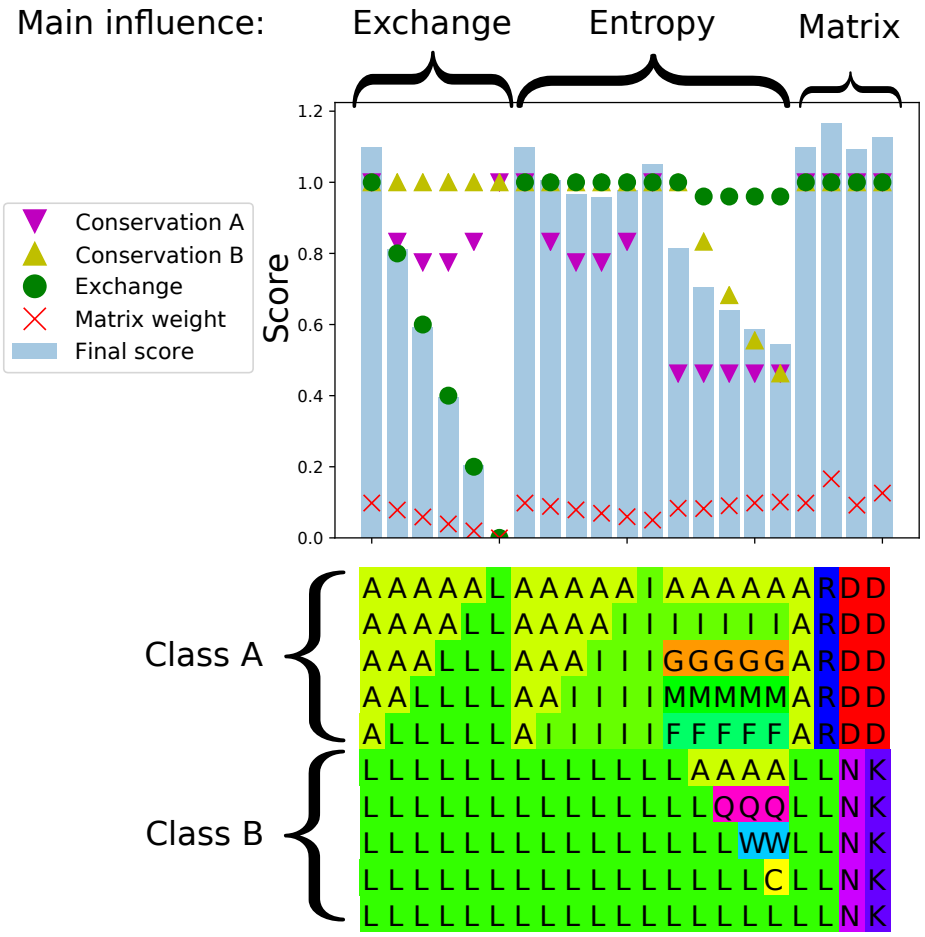


Figure 2: Influence of the different scores for a hypothetical alignment of two protein subfamily classes. On top the main influence is indicated: The exchange of an amino acid between the subfamilies (Exchange), the influence of the entropy inside a subfamily (Entropy) and the influence of an amino acid exchange of different types of amino acids (Matrix). The default parameters of SSR-viz were used for this example.

`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.

6 SSR plot

As soon as the alignment file and the class label csv file are created, the detection of significant SSRs can be performed. Therefore, the alignment and the CSV-file need to be loaded with the SSR_plot panel, if they are loaded correctly, an additional window will open, which allows to set the parameters for the detection algorithm and the output options.

6.1 Arguments

`Input CSV-file`

The CSV-file holding the class labels.

`Class label`

The name of the column holding the subfamily class label information, normally this should just be called "Class", but an alternative column name can also be chosen, in order to use alternative class definitions. See also: 9.

`Alignment`

The alignment file used to create the CSV-file. Different alignments can be chosen in order to test the influence of the alignment parameters or the alignment algorithm. The names of the sequences in the file need to match the names in the CSV-file.

`Verbose`

Shows some additional information about the loading process, for example which files do not match between CSV and alignment file. (Ideally this should not happen.)

7 SSR draw

In this new window all parameters for the algorithm and the output can be adjusted. Each time the start button is pressed the SSRs are computed based on the current parameters.

Most of the arguments are self-explanatory, therefore only the more advanced options will be explained in the following argument list.

7.1 Arguments

7.1.1 File

`File name`

In many cases it is useful to compute multiple plots and files in order to compare the output with each other. The computed files have the "file name" argument as prefix. Various runs can be performed successively by tuning the parameters and changing the file name for each run.

7.1.2 Algorithm

Here the basic parameters for the algorithm can be set, see also Section 4.

`Outliers threshold`

In order to detect the most significant positions a detection threshold can be set, based on the Z-score, see `https://en.wikipedia.org/wiki/Standard_score`.

`Best positions`

A more simple, but less significant approach to filter the most important SSRs can be done by simply setting a threshold for the best positions, this approach will always return a set of residues.

The Z-score and Best positions argument can also be set together.

`Gap importance`

The implemented algorithm considers gaps as a kind of special amino acid, they are assigned a positions in the replacement matrices. It is difficult to judge the importance of gaps for the detection of SSRs. For example, if a conserved position in protein class A, is exchanged with a gap in protein class B, this could be an important information or it could be due to the alignment algorithm. Therefore, we decided to let the user judge the gaps individually. If the "gap importance" is set to 0, an exchange with a gap will be considered as unimportant. If the "gap importance" is set to 1 it is considered as an important amino acid exchange. The influence of the gap can be judged by running the algorithm with both settings and checking if the significant positions are changed.

`Matrix`

This allows to choose the substitution matrix used for the algorithm (see 4), all matrices which are implemented in Biopython can be used, see

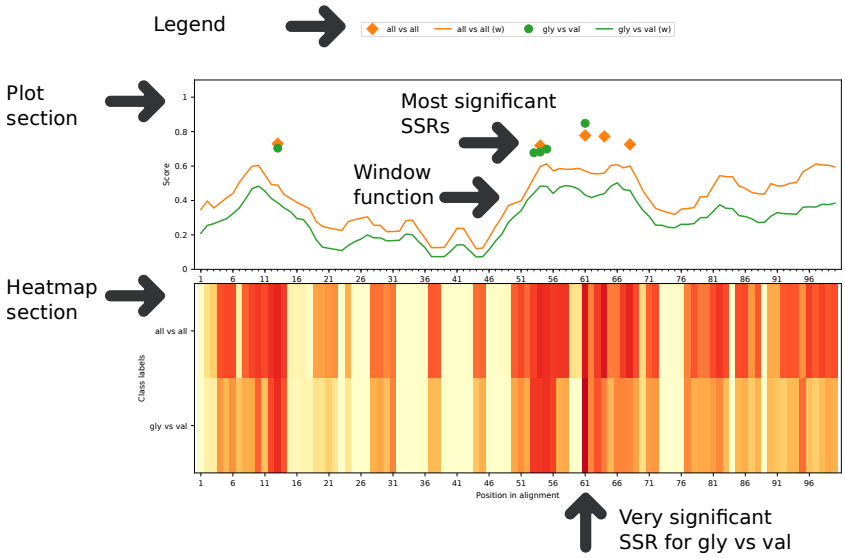


Figure 3: Example SSR-viz output: The SSRs are computed for the adenylation domain of the NRPS system. The subfamily classes are defined according to the substrate specificity of the domain: glycin (gly) and valine (val). See also the example in Section 9.

`http://biopython.org/DIST/docs/api/Bio.SubsMat.MatrixInfo-module.html` for details.

Additionally, we computed a substitution matrix purely based on the physico-chemical properties of the amino acids, based on [5].

`Weight of the replacement matrix`

The basic algorithm computes the most significant residues to distinguish between protein subfamilies. Amino acid properties are not considered for this analysis. In many cases even residues which are very similar can have a great influence on protein function, in other cases they have no influence at all. Therefore, we allow the user to decide how important the substitution matrix should be weighed. The default parameter of 0.01 is usually a good start. This will only slightly influence the initial order of the residues, but for those residues with the same score, they will be reordered with a higher focus on residues replaced with more different residues. See also Section 4 and Section 9.

7.1.3 Figure

The created visualization can be adjusted. The output figure consists of two parts, a heatmap of the SSRs and a plot with the most significant positions on top (see Figure 3).

Keep in mind that for projects with multiple protein sub-families the plots can be easily overcrowded. A maximum of 10 panels for the heatmap and 5 panels for the plot are recommended.

`Remove label from top of plot`

Usually a legend is plotted on top of the plots, this looks only nice for about 5 or 6 labels. It can be removed, but the recommended way is to plot the SSRs for less subfamilies.

`Figsize hight / Figsize width`

The default figure size is Din A4 but any other size can be chosen.

`Chunk size of the figure`

How many residues should be plotted per page.

`Entire alignment in one plot`

This argument would squeeze the entire alignment in one plot, which can be useful to get on overview of the alignment.

`Tick positions`

How often the positions should be shown in the bottom ticks.

`Ratio plot / ratio heatmap`

Sets the ratio between plot and heatmap.

7.1.4 Additional output

Additional to the created figure, two additional outputs can be created:

`Jalview annotation file`

This file can be imported to Jalview, so that the SSR scores can be visualized together with the alignment. Open the alignment with Jalview and click File, then Load Features/Annotations and choose the created *.jv_plot.txt

`Plot statistics`

This creates a csv file, which shows tabular information of the SSRs created. The file lists the significant positions for each scoring scheme, including the positions in the alignment, the score, the conserved residue and the entire variety of residues for each subfamily.

7.1.5 Plot Options

Here the plot output can be modified.

`Window size / Window type`

> In addition to the most significant positions, a window function can be applied to the SSRs. This window function can be used to visualize general areas of interest in the alignment. Possible windows are: mean, max, min, std. If the window is set to 1 it shows you the exact SSRs for each residue.

`No all vs all plot`

> Besides the chosen one-vs-one and one-vs-all schemes, the plot always shows the SSRs for the all-vs-all scheme, representing the residues which are most important to distinguish between all subfamilies. The scheme can be removed with this option.

7.1.6 One-vs-One plot

Here are all the subfamilies listed which were parsed from the subfamily class label CSV-file. Each subfamily can be chosen to compute the SSRs. Keep in mind, that an all-vs-all scheme can become very large, as each subfamily will be compared to each other. Best practice would be to concentrate on a few (3-4) subfamilies and create multiple plots.

7.1.7 One-vs-All plot

The choice is identical to the One-vs-one plot, but shows most significant SSRs to distinguish each subfamily from all the other subfamilies.

7.1.8 Heatmap Options

`No all vs all row in the heatmap`

> Besides the chosen one-vs-one and one-vs-all schemes, the heatmap always shows the SSRs for the all-vs-all scheme, representing the residues which are most important to distinguish between all subfamilies. This scheme can be removed with this option.

`Remove the labels from the heatmap`

> The labels can be removed from the heatmap. For example if many one-vs-one schemes are shown, it looks better without labels.

7.1.9 One-vs-One Heatmap

Identical to One-vs-One plot, but for the heatmap.

7.1.10 One-vs-all Heatmap

Identical to One-vs-All plot, but for the heatmap.

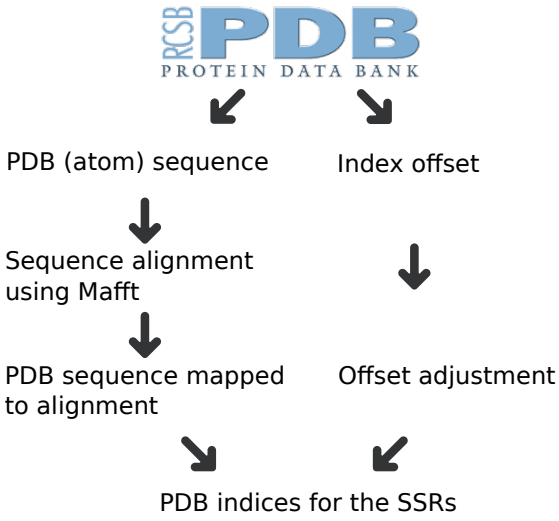


Figure 4: PDB mapping and offset adjustment scheme for the **Add_pdb** tool.

8 Add pdb

In order to simplify the analysis of the detected SSRs in a structural context, this tool allows to assign indices of a PDB file to the SSRs listed in the plot statistics CSV-file ('stats.csv'). The offset (PDB structures are often shifted relatively to the original protein sequence) is adjusted (see Figure 4).

Multiple PDBs can be used, if the `Input pdb file` argument is changed and the tool is applied multiple times. If the input 'stats.csv' and the output 'stats.csv' are identical each run will create a new row with PDB indices.

Note that if the mapping of the structure to the sequence leads to undesired results (for example in case of a multi protein complex MAFFT does not know which part to align) it might be necessary to trim the PDB file to only contain the protein part that should be used. This can easily be done with pymol and is demonstrated in the use-case (see 9)

8.1 Arguments

`Input sequence alignment`

Input file must be a sequence alignment in FASTA format (the one used to create the plot statistics CSV-file ('stats.csv')).

`Input pdb file`

Input file must be a PDB file which corresponds to the alignment (sequences should have a high similarity). A good output can only be expected from a structure with reasonable sequence similarity.

`Chain in the pdb file`

The chain in the pdb file, which should be aligned.

`Temporary folder`

Directory for the added alignment of the pdb and the MAFFT map file. This can also be the folder where the other outputs files are stored.

`Mafft executable`

Mafft is used internally to add a sequence to an alignment. In Linux if MAFFT is installed globally (it can be executed via the command line from anywhere), the default parameter should be fine. In Windows one might have to search for the correct executable path (see 10.1).

`Stats csv file`

The indices of the PDB are added to the 'stats.csv' file created with SSR_plot.

`Output csv`

Optional different output for the 'stats.csv' file, can be identical with the input file

9 Use case

In the following example the adenylation domain of the nonribosomal peptide synthetases (NRPS) was analyzed. The substrate specificity of this domain is responsible for the high diversity of secondary metabolites of the type of nonribosomal peptides, a fast source for bioactive molecules such as antibiotics and cytostatics. Further information can be found at: `https://en.wikipedia.org/wiki/Nonribosomal_peptide`.

All files needed to reproduce this example are provided in `https://github.com/PhaBiFreiburg/SSR-viz/use_case`.

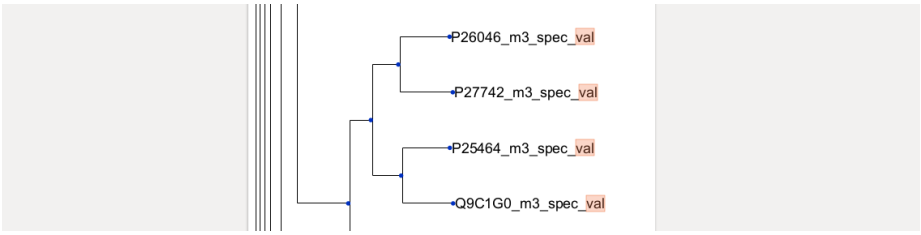
For this use case two subfamilies were chosen, with respective specificities for valine and glycine. For both subfamilies crystal structures are available (PDB: 3VNS, 4ZXI), so that the detected SSRs can be interpreted in a structural context.

The sequences and subfamily labels were taken from Rausch *et al.* [6], one of the first comprehensive analyses of this domain. The provided sequences were converted into a FASTA file using a simple Python Script. The original Data, the script and the FASTA file can be found in `use_case/raw/a_doms/`.

9.1 The clustering problem

The provided example demonstrates why in many cases a clustering or phylogenetic analysis will not lead to the a desired subfamily grouping. The sequences are labeled according to their substrate specificity, which makes it easy to investigate the results of any clustering algorithm.

(a) Cluster in accordance with specificity:



(b) Cluster not in accordance with specificity:

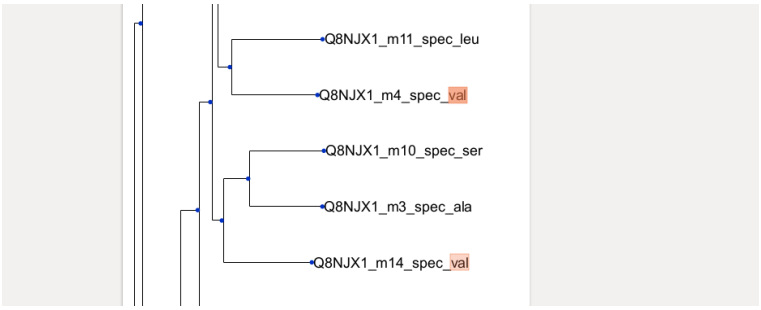


Figure 5: Example of a phylogenetically clustering for the adenylation domain. Whereas some clusters can be based on the specificity of the protein (a), others are rather due to the origin of the sequence (b).

```
>Q9FDB3_m2_spec_pro # <- label
```

The sequences were clustered using the default parameters of MAFFT.

```
MAFFT --auto NRPS_A_dom.fasta > NRPS_A_dom_ali.fasta
```

For this alignment a phylogenetic tree was generated using the web service of EMBL `https://www.ebi.ac.uk/Tools/phylogeny/simple_phylogeny/` with default parameters. The resulting tree can be found in `use_case/raw/a_doms/phylo/A_dom_tree.txt.pdf`. The careful observation reveals, that even though in some cases the sequences are clustered according to their specificity (see Figure 5a), in most cases the clustering is rather based on evolutionary distance (see Figure 5b).

Therefore, if the desired subfamily class label definition is not directly associated with the evolutionary distance of the sequences, an automated labeling approach will not lead to correct assignment. Which means, that for best results in SSR detection the subfamily class label needs to be extracted from the literature of experimentally derived labels.

Nevertheless, SSR-viz can also be used for SSR detection of phylogenetical clustered sequences, if this is the desired clustering.

9.2 CSV builder

Fortunately, for this example the subfamily class labels are already assigned and the sequences can be used to detect SSRs for the discrimination of sequences with specificity for for valine and glycine.

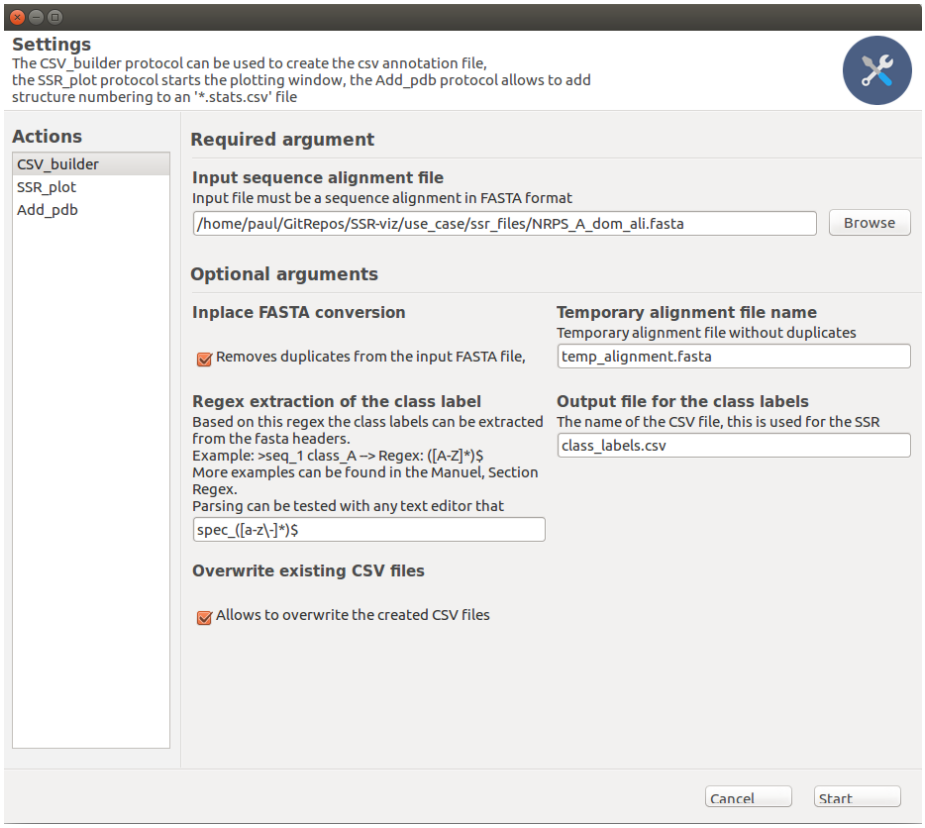


Figure 6: Example of parameters used to create the subfamily class label CSV-file.

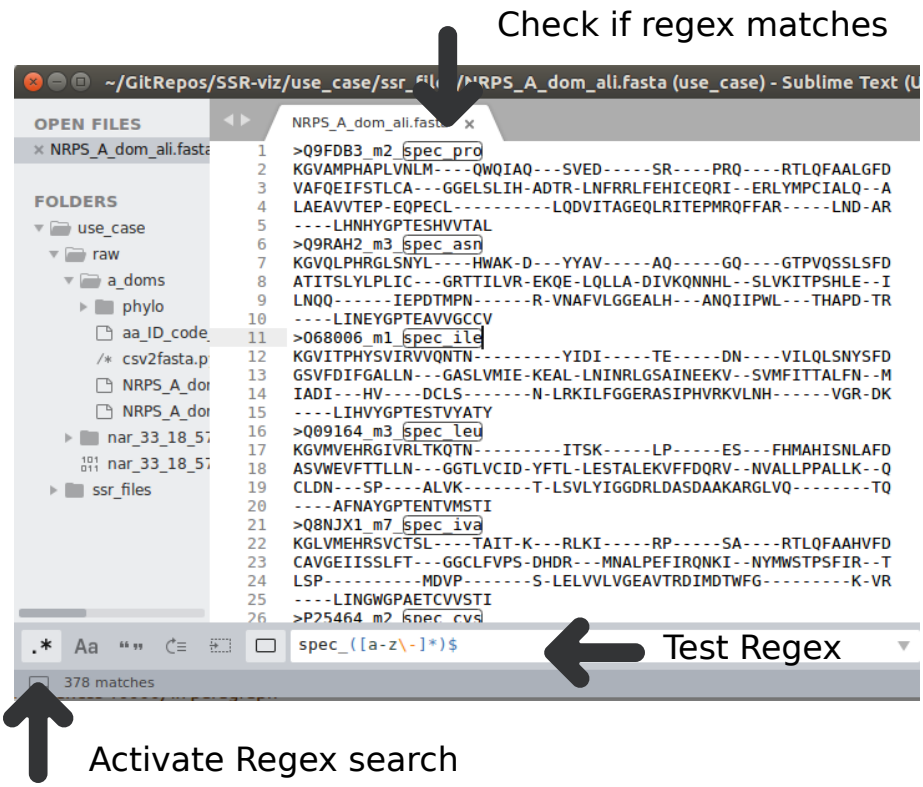


Figure 7: Usage of Sublime 3.0 to design the correct Regex for subfamily class label extraction.

A subfamily class label CSV-file needs to be created (see Figure 6).

This could be a file with blank class labels, which need to be manually assigned. Since the class label can be found in the FASTA sequence name tag, the Regex extraction routine can be used, which makes this task much easier.

```
>Q9FDB3_m2_spec_pro # <- label
```

To design a working Regex, the best option is to open the alignment in a text editor which supports Regex search and test if the class label can be found with the Regex string (see Figure 7).

It is important to create a Regex which can extract all possible labels. For example in this use case the label might contain the sign '-', so this needs to be included into the Regex.

```
>O33743_m1_spec_lys-b # <- label
```

The **CSV_Builder** will in any case report, if the label for some domains could not be found (see Figure 8). So in this example, a Regex which can also capture numbers would be better:

```
spec_([0-9a-z\-]*)$
```

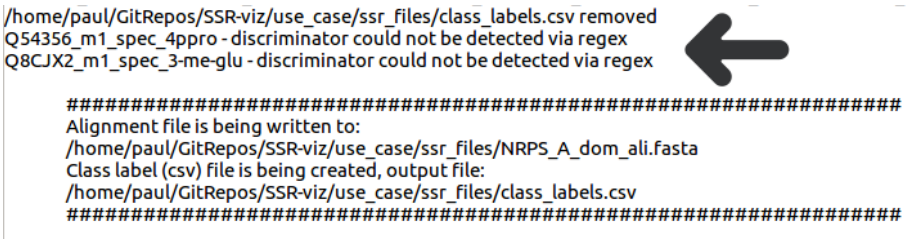


Figure 8: Example of labels, that could not be found with the regex, in this case, due to the missing number in the Regex.

	Name	Class
2	Q9FDB3_m2_spec_pro	pro
3	Q9R9J0_m4_spec_pro	pro
4	Q9RAH1_m2_spec_pro	pro
5	O05647_m1_spec_pro	pro
6	O30980_m1_spec_pro	pro
7	Q8NJX1_m13_spec_pro	pro
8	Q01886_m1_spec_pro	pro
9	Q44928_m1_spec_pro	pro
10	Q9RAH4_m3_spec_pro	pro
11	O07944_m1_spec_pro	pro
12	Q9KWY7_m1_spec_pro	pro
13	P94459_m1_spec_pro	pro
14	Q93I55_m4_spec_pro	pro
15	Q9L8H4_m1_spec_pro	pro
16	O30408_m1_spec_pro	pro

Figure 9: Example of the subfamily class label CSV-file.

If the label extraction was successfully, a file named class_labels.csv is created in the same folder as the MSA. This file contains the corresponding subfamily class label for each sequence (see Figure 9). This file can be adjusted, for example if labels should be merged, changed or alternative labels should be used. For alternative labels just add another column on the top row and add the desired labels (see Figure 10).

9.3 SSR plot

The MSA and class label CSV-file are now ready to be parsed by the **SSR_plot** tool (see Figure 11).

If the files can be parsed a new window opens **SSR_draw**.

9.4 SSR draw

In this window all parameters for the SSR detection and visualization can be adjusted (see Figure 12). In this example the default parameters are chosen, which should be a good start for most analysis.

In order to demonstrate the full range of output options the Jalview Annotation File as well as the plot statistics file should be chosen as additional output (see Figure 13).

	A	B	C
1	Name	Class	Class_2
2	P94873_m1_spec_aad	aad	acidic
3	Q01757_m1_spec_aad	aad	acidic
4	P25464_m1_spec_aad	aad	acidic
5	P27742_m1_spec_aad	aad	acidic
6	Q9RLP6_m4_spec_alaninol	alaninol	hydrophil
7	Q09164_m1_spec_ala-d	ala-d	aliphatic
8	Q09164_m3_spec_leu	leu	aliphatic
9	Q8NJX1_m11_spec_leu	leu	aliphatic
10	Q08787_m1_spec_leu	leu	aliphatic

Figure 10: In this example the alternative label assigns the amino acids to a higher hierarchical definition. This could be used to find SSRs responsible to discriminate proteins with specificity for aliphatic from proteins with specificity for hydrophobic amino acids.

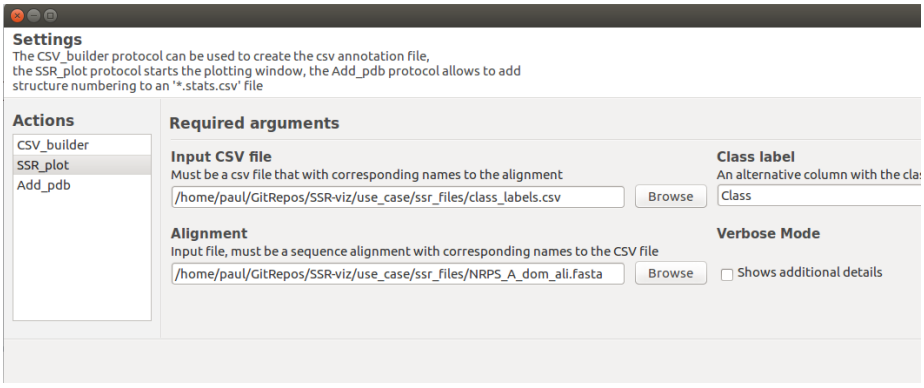


Figure 11: Example of parameters used to parse the MSA and class label CSV-file.

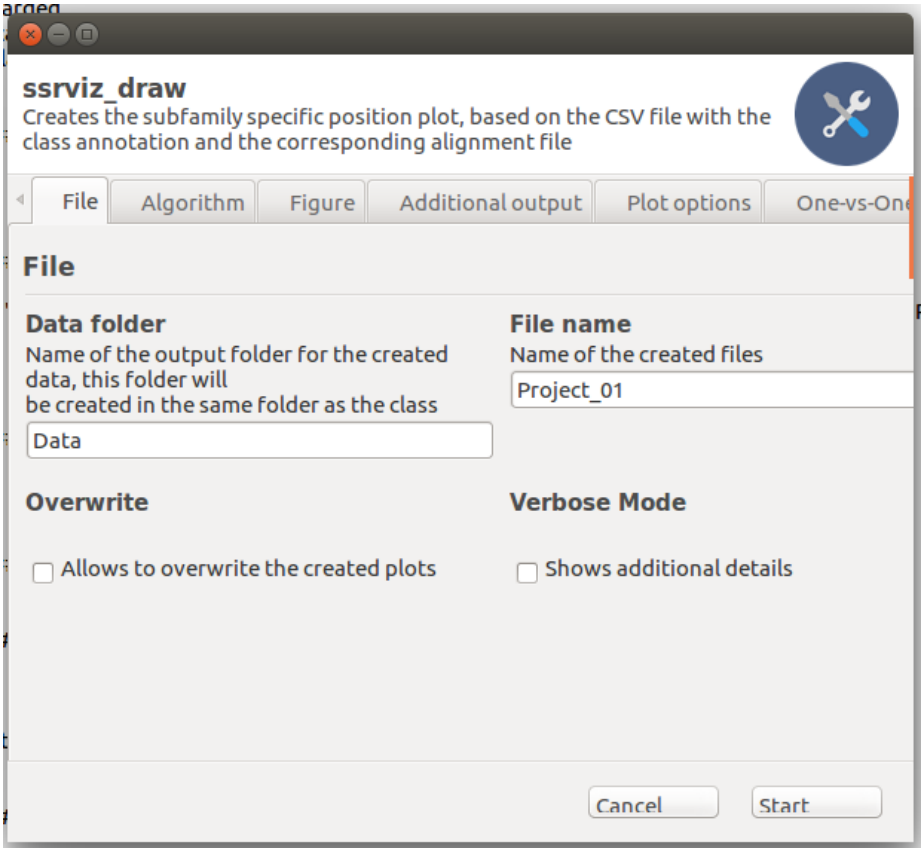


Figure 12: Example of the **SSR_draw** window.

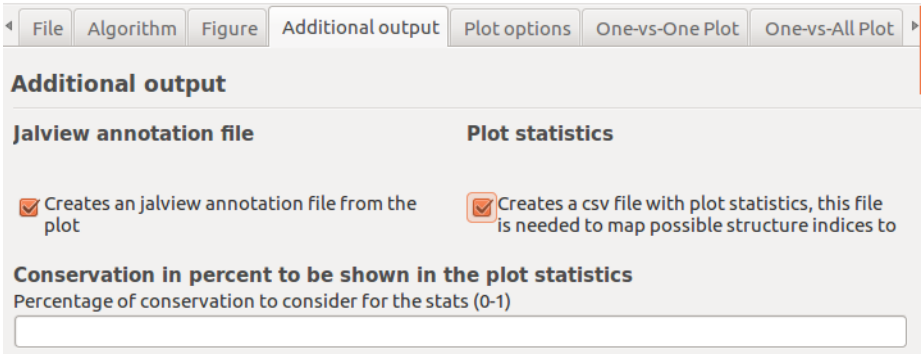


Figure 13: Example of the additional output option.

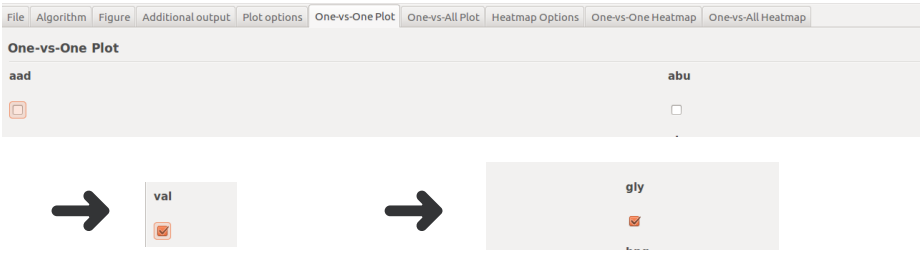


Figure 14: Example of the label choice for valine and glycine specificity

In order to visualize the differences for valine and glycine specificity those two subfamilies should be used as options for the one-vs-one plot and heatmap options (see Figure 14). Just scroll to the label and tick it. Of course any other label could also be used.

Start the program by clicking Start and if the program completes successfully, a new result folder named *Data* is created in the same folder as the MSA.

9.5 Output

In the results folder should be a plot called *Project_ 01.pdf*, which shows the results of the SSR detection in form of a plot and a heatmap as described in Figure 3. This plot can be used to gain an overall impression of the SSRs for the chosen subfamilies.

In this example, the SSRs for two subfamilies are shown alongside the SSRs for all the subfamilies. Some positions overlap, for example position 61 is very important to distinguish all the subfamilies from each other as well as the two subfamilies (this position is probably in the center of the active cite !). Other positions are only important for the two subfamilies. And again some positions are generally important for the substrate choice, but are of low significance for these two families. Additionally, the plot of the window function highlights general areas of interest. The first page of the plot is always a summary of the used parameters as shown in Figure 15.

In many cases it is useful to observe the detected SSRs in the context of the alignment, therefore the file 'Project_01_jv_plot.txt' is created, which can be loaded into the Jalview alignment editor (see Figure 16). This allows to investigate the SSRs together with the vast toolbox of the Jalview editor.

A summary of the mot significant SSRs can be found in the plot statistics CSV-file 'Project_01_stats.csv'. This file shows the most significant SSRs for all the schemes chosen in the plot options (see Figure 17). Additionally the conserved residue (Cons) of each subfamily for the SSRs are shown, as well as the full choice of residues (All res.).

9.6 Add pdb

If structures of the observed proteins are available, it is often desired to investigate the detected SSRs in a structural context. Theoretically, this could

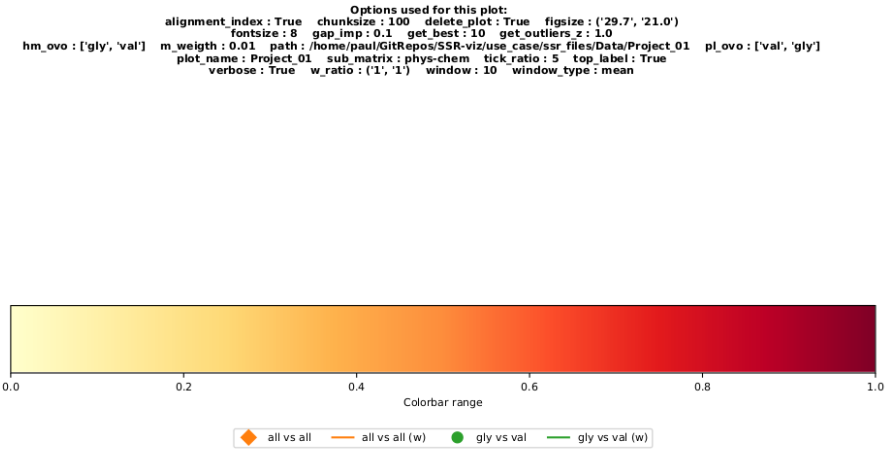


Figure 15: Example of the parameters page for the use case.

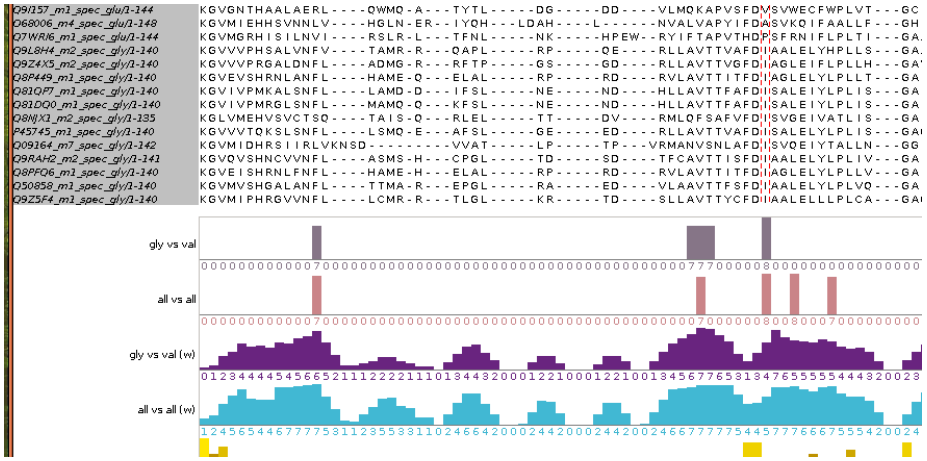


Figure 16: Example of the Jalview annotation file loaded into Jalview, together with the original alignment. Position 61 is marked. The proteins with specificity for glycine, show a conserved isoleucine at this position.

	A	B	C	D	E	F
1	gly vs val					
2	Position	Score	gly Cons	gly All res.	val Cons	val All res.
3	61	0.848277469	I	I	A	AFMV
4	197	0.7477964433	W	LQW	F	CFLY
5	116	0.7366427768	W	LW	F	AFLV
6	110	0.7249220942	Q	MQ	W	FLMSW
7	13	0.704088194	F	FLS	L	FLS
8	55	0.6985482121	T	ST	S	AGS
9	54	0.6815126335	V	FV	T	AFHILT
10	196	0.678884851	I	IMV	T	ITV
11	53	0.6774003302	A	ANQ	F	FHLMQ
12	147	0.6715864704	G	GLR	L	LV
13	all vs all					
14	Position	Score	leu Cons	leu All res.	val Cons	val All res.
15	197	0.8029639576	V	ACDFMV	F	CFLY
16	110	0.7876686718	F	AFLMY	W	FLMSW
17	61	0.7863987456	A	AFG	A	AFMV
18	64	0.7816837075	W	AFILW	F	AEFGLWY
19	188	0.7786743914	N	ACEGLMN	G	ACGV
20	115	0.7477713738	L	FILMSV	L	FLQTV
21	13	0.741727644	L	CFLNT	L	FLS
22	68	0.7319081458	P	AFGMPTW	G	AGLPV
23	54	0.729066302	L	FHILMVY	T	AFHILT
24	151	0.7279836069	L	FLMTY	I	FILTW
25						

Figure 17: Example of the plot statistics CSV-file. The top 10 SSRs are shown for subfamilies with specificities for glycine and valine, as well as all the subfamilies. Whereas many positions overlap, position 116 is only of high relevance for the two subfamilies.

be achieved by using Jalview, as it supports the inclusion of structures to the MSA, but many users prefer a sophisticated protein viewer like pymol, which requires the mapping of any structure to the MSA. Therefore, a simple routine was implemented which allows to map the indices of a PDB file to the SSRs in the plot statistics CSV-file.

For this example the PDBs of a member of the subfamily with glycine specificity (PDB: 4zxi) as well as valine specificity (PDB: 3vns) were used. Additionally, the structure (PDB: 1amu) was added, as this is the structure where Stachelhause *et al.* [7] based their investigation upon, which allows an easy comparison of the detected SSRs with the proposed specificity-conferring code. An example of the parameters used to map the indices are shown in Figure 18.

The indices of the structure are added to the plot statistics CSV-file (see Figure 19).

9.7 Results

When all the output files are created and structure indices are added the detected SSRs can be investigated. Lets have a look at the SSRs detected for the two subfamilies. This can for example be achieved by loading the PDB into pymol and selecting the SSRs with a command such as:

```
select 3vns and resi 209+203+202+201+315+314+274+183+257+251
```

The SSRs detected for each subfamily are shown in the respectively structure in Figure 20 and Figure 21. All SSRs are in close proximity to the substrate as expected for the specificity defining residues. Interestingly, only 4 residues directly interact with the substrate. The overlap of the SSRs for both subfamilies

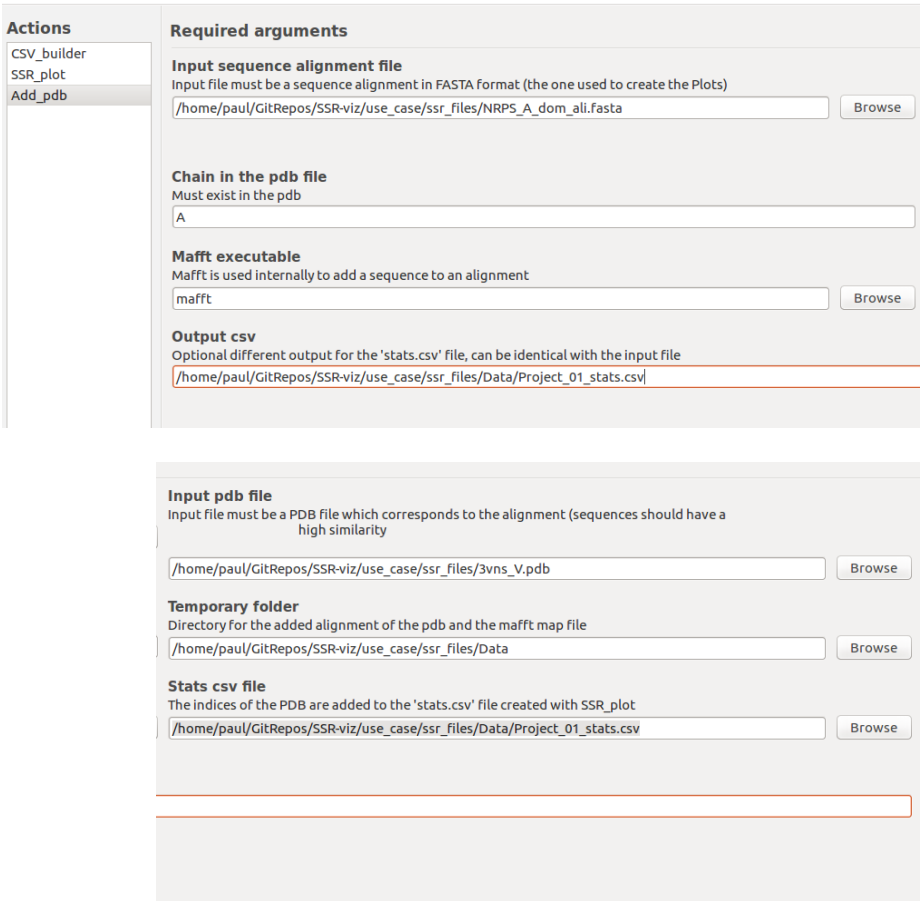


Figure 18: Parameters used to map the indices of 3vns to the MSA.

gly vs val

Position	Score	gly Cons	gly All res.	val Cons	val All res.	1amu	3vns_V	4zxi_G
61	0.848277469	I	I	A	AFMV	236	209	671
55	0.6985482121	T	ST	S	AGS	230	203	665
54	0.6815126335	V	FV	T	AFHILT	229	202	664
53	0.6774003302	A	ANQ	F	FHLMQ	228	201	663
197	0.7477964433	W	LQW	F	CFLY	331	315	769
196	0.678884851	I	IMV	T	ITV	330	314	768
147	0.6715864704	G	GLR	L	LV	295	274	732
13	0.704088194	F	FLS	L	FLS	210	183	645
116	0.7366427768	W	LW	F	AFLV	284	257	719
110	0.7249220942	Q	MQ	W	FLMSW	278	251	713

Figure 19: PDB indices added to the plot statistics CSV-file

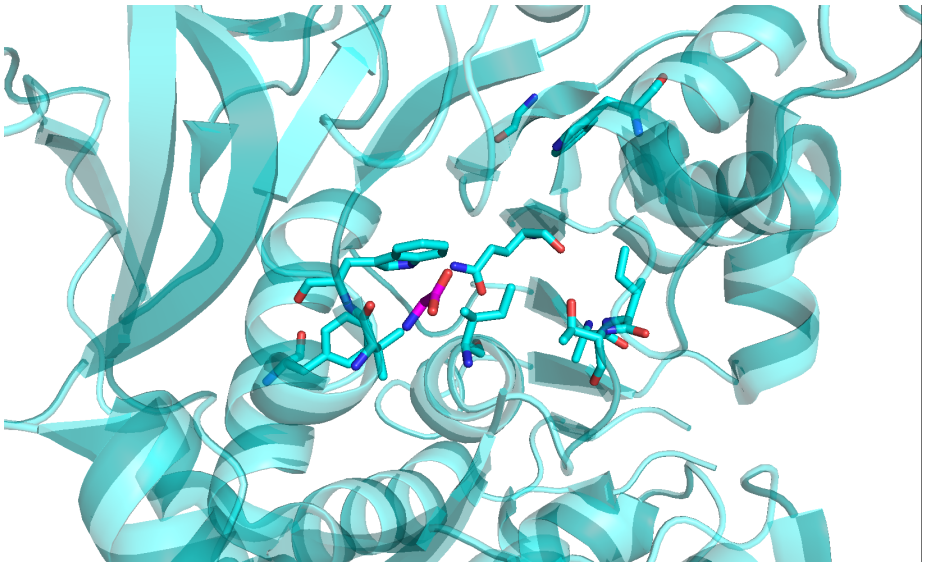


Figure 20: SSRs for 4zxi with specificity for glycine

are shown in Figure 22. Position 331 and 236 (based on 1amu numbering) are the most significant SSRs for those two subfamilies, which can be explained trough steric interaction with the substrate.

The SSRs for the all-vs-all scoring scheme can be compared with the observations of Stachelhause *et al.* [7] as the most significant SSRs for all the subfamilies should correlate with the proposed specificity-conferring code.

In the proposed specificity-conferring code the residues are divided into highly variant and moderately variant residues. Of the detected SSRs all the highly variant residues are included in the top 10 SSRs (239, 278, 299, 322, 331 (based on 1amu numbering)).

9.8 Conclusion

The detected SSRs can explain the substrate selectivity of the investigated protein subfamilies. Also, the SSR analysis of the entire family is in good accordance with proposed position of the literature, demonstrating the general applicability of SSR-viz.

Feel free to further investigate the provided adenylation domain example. For example, it can be analyzed how the subfamilies with specificity for aliphatic amino acids are distinguished from each other. Or the One-vs-All scheme can be explored by investigating how any subfamily is distinguished from all the others.

If you have your own protein subfamily research project feel free to explore the SSRs of a given problem using the entire range of SSR-viz.

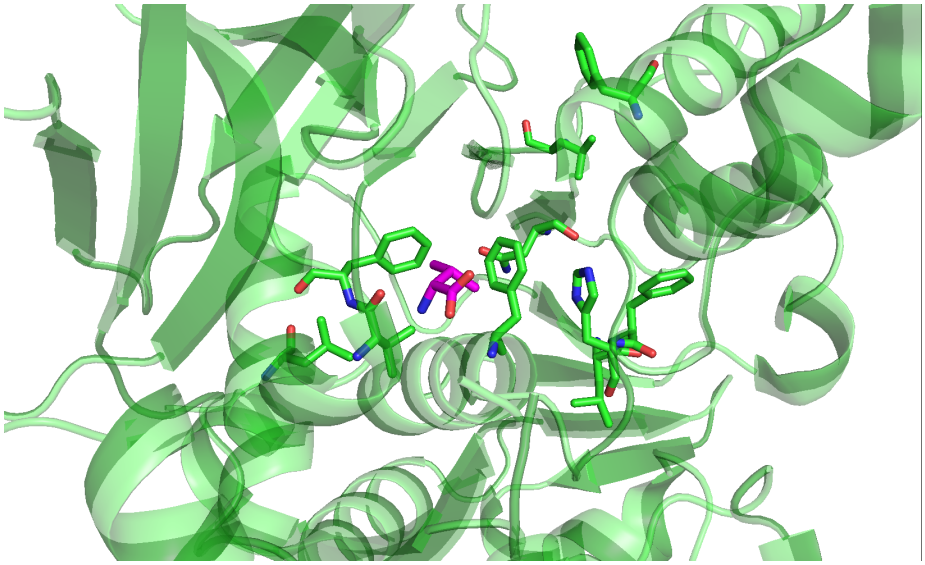


Figure 21: SSRs for 4zxi with specificity for valine

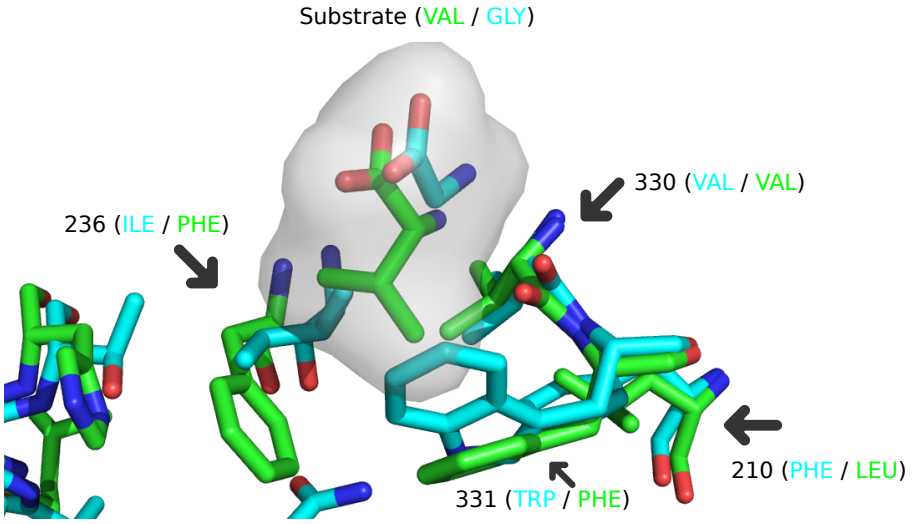


Figure 22: Overlap of the SSRs in close proximity to the substrate.

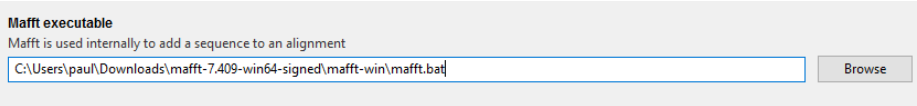


Figure 23: Example of the MAFFT executable path.

10 Appendix

10.1 How to install and set up Mafft

Mafft is needed as the alignment tool used to map the PDB indices to the MSA. Therefore, SSR-viz can in general work fine without MAFFT, if the **Add_PDB** tool is needed MAFFT needs to be installed.

10.1.1 Linux

For Linux MAFFT can be installed very easy, just type

```
sudo apt install MAFFT
```

Or check on their homepage `https://MAFFT.cbrc.jp/alignment/software/` for other installation options.

If the installation is done, one should be able to call MAFFT with:

```
MAFFT input > output
```

In this case the default MAFFT executable path should be fine. If MAFFT is stored in another place, the path needs to be adjusted.

10.1.2 Windows

In windows the tested way to get MAFFT working is to install MAFFT as described in `https://MAFFT.cbrc.jp/alignment/software/windows_without_cygwin.html`. As this is an all-in-one package, one basically only needs to download the .zip file and unpack it and MAFFT is up and running.

To use MAFFT in SSR-viz, the 'MAFFT.bat' file needs to be selected as MAFFT executable path (see Figure 23). To simplify the usage, it is meaningful to unpack the MAFFT .zip file inside the SSR-viz folder.

10.2 Regex examples

If sequences are download from the literature or created from other sources using a script, they probably contain the subfamily class label information somewhere in their FASTA name tag. Here are some examples of possible names and how to extract the subfamily class label using regex. Keep in mind that brackets () are used to define the part which should be extracted.

Seq. name	Regex	Label
>sequence1_spec_ADP >sequence2_spec_ATP	`_spec_([A-Z]*)$`	ATP ADP
>sequence1_1 >sequence2_2	`_([0-9]*)$`	1 2
>ASH1L-1_Class:0 >ATAD2-1_Class:1	`Class:([0-9]*)$`	0 1
>Q9FDB3_m2_spec_pro >Q00869_m1_spec_hyv-d	`_spec_([A-Z\-]*)$`	pro hyv-d
>AT#Alpha#0030#m >AT#Alpha#0020#mm	`\#([a-z]*)$`	m mm
>AB-sequence1 >AK-sequence1	`^([A-Z]*)\-`	AB AK

References

[1] F. Sievers and D. G. Higgins. "Clustal omega". *Current Protocols in Bioinformatics* 48 (12, 2014), pp. 3.13.1–16.

[2] J. D. Hunter. "Matplotlib: A 2D Graphics Environment". *Computing in Science Engineering* 9 (2007), pp. 90–95.

[3] M. Clamp *et al.* "The Jalview Java alignment editor". *Bioinformatics* 20 (12, 2004), pp. 426–427.

[4] D. W. Mount. "Using BLOSUM in Sequence Alignments". *Cold Spring Harbor Protocols* 2008 (1, 2008), pdb.top39.

[5] C. Chrysostomou and H. Seker. "Novel protein weight matrix generated from amino acid indices". In: *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). 2015, pp. 8181–8184.

[6] C. Rausch *et al.* "Specificity prediction of adenylation domains in nonribosomal peptide synthetases (NRPS) using transductive support vector machines (TSVMs)". *Nucleic Acids Research* 33 (2005), pp. 5799–5808.

[7] T. Stachelhaus, H. D. Mootz, and M. A. Marahiel. "The specificity-conferring code of adenylation domains in nonribosomal peptide synthetases". *Chemistry & Biology* 6 (1999), pp. 493–505.