BioShell 3.0

Release 3.0

May 16, 2019

Overview

CHAPTER 1

Introduction

1.1 BioShell applications

BioShell is **a set of command-line programs** for easy data manipulation from a UNIX-like terminal or a shell script. The programs can read and write standard file formats and handle protein sequences and structures. The tools helps also in simple calculations, like sequence alignment, Phi/Psi angles, crmsd and many more. See *Programs page* for details.

1.2 BioShell tests & examples

Since the most recently published version 3.0, BioShell package comes with **extensive set of example applications**, which have been created to simultaneously reach tree goals:

- **to extend the set of BioShell command line tools**. Programs with names starting with `ap_` are in fact yet another applications. The difference between these test and *standard* apps is that the latter perform only a single action and their command line is simplified. These programs are integration tests at the same time.
- **to provide high quality code snippets that help BioShell users write their own programs**. Small programs, that show how to use a particular class or a function, are named `ex_*`. At the same time they serve as *unit tests*
- **to test the code**. Both `ex_*` and `ap_*` tests are automatically executed by a test server to ensure the quality and integrity of the package. Input data as well as curated output of these tests is versioned in git repository along the source code.

All the examples are included in respective API documentation pages. Since the test are continuously tested, the serve as a source of validated snippets for creating future programs.

1.3 BioShell library for Python (aka PyBioShell)

BioShell distribution provides also bindings to Python scripting language; that is, BioShell is also a **versatile library for python scripting.** BioShell objects can be imported as any other python modules. Example scripts are also included in the repository.

Precompiled library (a single `.so` file) for Unix distribitions can be downloaded from this page. The compilation process is described here

1.4 BioShell C++ library

Finally, BioShell is **a C++ software library**. Both `ap_*` and `ex_*` BioShell tests are included in respective API documentation pages. Since the test are continuously tested, they serve as a source of validated snippets for creating future programs.

1.5 Previous versions

1.5.1 BioShell versions 1.x

The original BioShell package was designed as a suite of programs designed for pre- and post-processing in protein structure modeling protocols. The package has been providing a convenient set of tools for in conversion between various sequence and structure formats. It has been also possible to calculate simple properties of protein conformations. The very first commands (e.g. HCPM for clustering protein structures) were implemented in C, later on the development switched to C++.

1.5.2 BioShell versions 2.x

Around 2006/07 BioShell has been reimplemented in JAVA, designed as a library for scripting languages running on Java Virtual Machine, most notably Python, but also Scala, Ruby, Groovy and many others. Currently the most recent stable release is 2.2. API docs as well as example scripts may be found in documentation. All program from 1.x versions were also ported to JAVA.

1.6 Citations

- **BioShell - the third version:** J.M. Macnar, N.A. Szulc, A.E. Badaczewska-Dawid and D. Gront *"Exhaustive tests set of BioShell 3.0 suite for structural bioinformatics"* Bioinformatics *submitted*
- **Three-dimensional protein threading:**

  D. Gront, M. Blaszczyk, P. Wojciechowski, A. Kolinski *"Bioshell Threader: protein homology detection based on sequence profiles and secondary structure profiles."* Nucleic Acids Research **2012** doi:10.1093/nar/gks555
- **One-dimensional protein threading:**

  P. Gniewek, A. Kolinski, D. Gront *"Optimization of profile-to-profile alignment parameters for one-dimensional threading."* J. Computational Biology **2012** Jul;19(7):879-86
- **BioShell - the second version:**

D. Gront and A. Kolinski *"Utility library for structural bioinformatics"* Bioinformatics **2008** 24(4):584-585

- **BBQ - program for backbone reconstruction:**

  D. Gront, S. Kmiecik, A. Kolinski *"Backbone Building from Quadrilaterals. A fast and accurate algorithm for protein backbone reconstruction from alpha carbon coordinates."* J. Comput. Chemistry **2007** 28(9):1593-1597

- **BioShell - the first version:**

  D. Gront and A. Kolinski *"BioShell - a package of tools for structural biology computations"* Bioinformatics **2006** 22(5):621-622

- **Program for clustering protein structures (currently named *clust*):**

  D. Gront and A. Kolinski *"HCPM - program for hierarchical clustering of protein models"* Bioinformatics **2005** 21(14):3179-3180

CHAPTER 2

Installation

BioShell is written in C++11 and must be built before use. This is a quite easy process, which requires CMake (https://cmake.org) and a relatively modern C++ compiler such as gcc 5.0 or clang 10.0 The compilation procedure is as follows:

1. Install `zlib`

BioShell requires zlib library so it can handle compressed files. You must install developer version of the library to be able to compile BioShell. On Ubuntu linux it can be installed by the command:

```
sudo apt-get install zlib1g-dev
```

2. If you haven't done it yet, **clone bioshell-release repository** (https://bitbucket.org/dgront/bioshell-release/src/master/) from Bitbucket:

```
git clone git@bitbucket.org:dgront/bioshell-release.git
cd bioshell-release
```

This should create `bioshell-release` directory in your current location. The second line steps into this new directory

3. **Run CMake**:

```
cd build
cmake ..
```

The `build` directory will contain compilation intermediate files and may be deleted once BioShell is compiled. The first line enters that direcotry, the second command calls `cmake` to set up the compilation process. CMake attempts to set up everything automatically, sometimes however it would require some guidance, e.g. to find the right compiler (see below)

4. **Run Make**:

```
make -j 4
```

where `-j 4` allows make use 4 cores to run parallel compilations. This command will attempt to compile **all targets**; the list of all targets can be printed by `make help`. As one can see, each executable is a separate target. There are also predefined group targets:

bioshell compiles only *bioshell* library

bioshell-apps compiles *bioshell* library and bioshell toolkit applications, such as **seqc** and **strc**

examples compiles all examples, i.e. all **ap_** and **ex_** application

5. **Additional parameters for compilation**

The procedure described above compiles the package with the default settings: *Release* build with no profiling. To change it, you should **remove everything from** `./build` **directory** and generate new makefiles with new settings:

- in order to use a compiler other that the default one (e.g. gcc version 4.8), say:

```
cmake -DCMAKE_CXX_COMPILER=g++-4.8  -DCMAKE_C_COMPILER=gcc-4.8 -DCMAKE_
↪BUILD_TYPE=Release ..
```

or to use `icc` for instance:

```
cmake -DCMAKE_CXX_COMPILER=icc  -DCMAKE_C_COMPILER=icc -DCMAKE_BUILD_
↪TYPE=Release ..
```

- to brew a **debug** build, turn `-DCMAKE_BUILD_TYPE=Release` into `-DCMAKE_BUILD_TYPE=Debug`. So to make a **debug** build **without changing the compiler**, say just:

```
cmake -DCMAKE_BUILD_TYPE=Debug ..
```

- to make a profiling build (-pg option) for gcc or *Xcode Instruments* (https://developer.apple.com/library/ios/documentation/AnalysisTools/Reference/Instruments_User_Reference/TimeProfilerInstrument/TimeProfilerInstrument.html) add `-D PROFILE=ON` to the `cmake` command.

6. **Using IDE**

In the above examples, `cmake` was used to produce makefiles for to compile BioShell. `cmake` command may be also used to generate project files for other environments, in particular:

- to produce *.xcodeproj file for xcode:

```
cmake  -DCMAKE_BUILD_TYPE=Release -G Xcode
```

- or to prepare *solution* files for Microsoft Visual Studio (must be run on a Windows machine):

```
cmake  -DCMAKE_BUILD_TYPE=Release -G "Visual Studio 2013"
```

CHAPTER 3

PyBioShell Installation

PyBioShell is a set of Python bindings to **BioShell** library. It allows use of BioShell classes like any other Python modules. The closest tool similar by functionality is Biopython, which however is partially written in Python.

The easiest option to get PyBioShell on your machine is to download precombiled library, available for Python3.5 and Python3.7 from this page

Another way is to compile it from sources, following the steps given below. The procedure assumes your `bioshell-release` repository is located in `src.git/bioshell-release/` and `binder` in `src.git/binder/`; these paths are arbitrary but the commands must be adjusted accordingly

3.1 0. Prequisities

In order to compile `binder`, you need to have `Ninja` building tool (website) and cmake. You will also need python headers, available from `python-dev` package or similar (e.g. `python3.5-dev`). On Ubuntu Linux you can install them with `apt-get`:

```
sudo apt-get install ninja-build  cmake python-dev
```

The use of `clang` compiler is advised. Try to get `clang-6.0` or newer (see this link)

3.2 1. Clone and compile `binder`

To clone binder from its *github* repository:

```
git clone https://github.com/RosettaCommons/binder
cd binder
python3 ./build.py -j 4
```

where the last command actually builds *binder* using four CPU cores for that. Note, that *binder* uses more than 1GB of disc space and its compilation may take a few hours.

3.3 2. Build PyBioShell

Open `BuildPython.py` file and edit variables, adapting it to your system. In particular, you most likely have to fix `clang++` version (`LINKER_CMD` variable) as well as the path where the `binder` executable is located (`BINDER_PATH` variable) Make a directory `build_bindings` in the main BioShell directory, i.e in the directory where `pybioshell.config` is located. There are three scripts availble for other Python versions Bulid-Python27.py, BuildPython35 and BuildPython37.py. Choose your Python version and run the compilation as follow:

```
python BuildPython37.py
```

You should find your compiled version in `bin/pybioshell.so`. If you have any problems with compilation, please do not hesitate to contact us.

CHAPTER 4

BioShell programs

Currently, BioShell distribution provides the following programs:

seqc (*cookbook recipes*): sequence converter : a utility to convert between sequence data formats

strc (*cookbook recipes*): structure converter : a utility to work with PDB files

str_calc (*cookbook recipes*): structure calculator; perform various calculations on a PDB file

clust (*cookbook recipes*): calculates hierarchical clustering of arbitrary objects based on a map of pairwise distances between them

hist (*cookbook recipes*): simple utulity to make 1D and 2D histograms

Now you can browse *BioShell cookbook*, or read tutorials, listed below

4.1 `clust` tutorial : clustering sequences and structures

Clustering procedure allows one to divide arbitrary number of objects into groups accordint to their mutual (dis)similarity. This method is widely used in bioinformatics and molecular modeling to deal with data sets that are too large to be inspected manually. Here we give two examples of Hierarchical Agglomerative Clustering with BioShell package:

1) to cluster a pool of protein sequences
2) to cluster results of protein-peptide docking

The BioShell procedure for clustering divides the task into three steps:

1) **calculate a matrix of distances between elements subjected to a clustering analysis.**

   As a result, a flat text file should be produced. The three columns of that file must provide i-th element ID, j-th element ID and the respective distance value

2) **run the actuall clustering procedure.**

   Although the procedure can be stopped at a particular cutoff distance, we advise to conduct the calculations i.e. until all the objects are merged into a single cluster. Clustering tree will be stored in an output file

3) **analyse the clustering tree to retrieve clusters at a desired cutoff level**

Below we show how to perform these three steps for two different clustering applications

4.1.1 Example 1. Clustering protein sequences by their mutual sequence identity

Step 1: Calculating the distances

Clustering procedure should merge close sequences (i.e. small mutual distance) into a single cluster, while dissimilar sequences should be placed in different clusters. Unfortunately, sequence identity value (seq_id) cannot be used here because its largest value (1.0) denotes identical sequences. Here we propose to use 1.0 - seq_id as a distance function.

First we use `ap_PairwiseSequenceIdentityProtocol` program to evaluate all pairwise distances:

```
ap_PairwiseSequenceIdentityProtocol inp.fasta 8 0.4  > seq_id.out 2>LOG
```

where `inp.fasta` is the input file (FASTA format), `8` is the number of cores (threads run in parallel) and `0.4` is the smallest seq_id value to be written to a file.

Then the seq_id values are converted into distances with `awk` command line tool:

```
awk '{print $1,$2,1.0-$3}' seq_id.out > distances.out
```

Step 2: Clustering the data

Then we run the `clust` tool:

```
clust -in::file=distances.out \
    -n=46621 \
    -complete \
    -clustering:missing_distance=1.1 \
    -clustering:out:tree=tree-complete >clust_out 2>clust_log
```

The `-n` option is necessary to provide the number of objects subjected to clustering (not the number of distance values!). `-clustering:missing_distance` Provides the default distance value for the cases it's undefined. The clustering tree will be stored in a file specified by `-clustering:out:tree` option

Step 3: Analysis

```
clust  -in::file=distances.out \
    -n=46621 \
    -clustering:in:tree=tree-complete \
    -clustering:out:clusters \
    -clustering:out:distance=0.4 \
    -clustering:out:min_size=1
```

4.1.2 Example 2. Clustering results of protein-peptide docking

The input data set contains 12500 conformations of a protein receptor (1jd4) with a short peptide bound to its surface. The conformations were calculated with FlexPepDocking program from Rosetta modelling suite.

Step 1: Calculating the distances

Step 2: Clustering the data

We run *clust* program as above, just should remember to put the correct imput file name and to change the number of data elements (i.e. protein conformations)

```
clust -in::file=1jd4-pep-crsmd \
  -n=12500 \
  -complete \
  -clustering:missing_distance=15.1 \
  -clustering:out:tree=tree-complete >clust_out 2>clust_log
```

Step 3: Analysis

```
clust -in::file=all_vs_all_crmsd_15 \
  -n=12500 -clustering:out:clusters \
  -clustering:out:distance=2.5 \
  -clustering:out:min_size=10 \
  -clustering:in:tree=tree-complete
```

4.2 BioShell cookbook

This cookbook provides a bunch of handy one-liners that simplify daily tasks in structural bioinformatics.

4.2.1 `bash`-only recipes

Combine a bunch of `.pdb` files into a single multimodel-pdb:

```
k=0;
for i in *.pdb; do
    k=$(($k+1));
    echo "MODEL     "$k;
    cat $i;
    echo "ENDMDL";
done > all-pdb
mv all-pdb all.pdb
```

4.2.2 1. `seqc` recipes

1.1 Create FASTA from PDB (prints FASTA on a screen):

```
seqc -in:pdb=2gb1.pdb -out:fasta
```

1.2 Create FASTA from PDB, including secondary structure:

```
seqc -in:pdb=2gb1.pdb -out:fasta -in::pdb::header -out:fasta:secondary
```

Secondary structure annotation is extracted from the PDB file header (`-in::pdb::header` option is necessary to parse it)

1.3 Create SS2 file from PDB:

```
seqc -in:pdb=2gb1.pdb -out:ss2 -in::pdb::header
```

As above, the secondary structure is extracted from the PDB file header; all the probability values (last three columns in a SS2 file) are set either to 1.0 or 0.0

1.4 Count secondary structure elements in a bunch of PDB files, create a nice table:

```
for i in 2gb1.pdb 2fdo.pdb 1rrx.pdb
do
  ss=`seqc -in:pdb=$i -out:ss2 -in::pdb::header -of -out::sequence::width=0 \
     | tail -1 | fold -w1 | uniq | sort | uniq -c | tr '\n' ' '`
  echo $i $ss
done 2>/dev/null
```

As in **recipe 1.2**, but this time a combination of a few bash commands is used to parse the ouput and count the number of secondary structure elements: coil (**C**), strands (**E**) and helices (**H**). Example output looks as below:

```
2gb1.pdb 6 C 4 E 1 H
2fdo.pdb 7 C 6 E 3 H
1rrx.pdb 16 C 11 E 5 H
```

1.5 Write FASTA file with only one line per sequence (un-wrap sequences)

```
seqc -in:fasta=in.fasta -out:sequence:width=0 -out:fasta
```

1.6 Convert ASN.1 sequence profile (*psiblast* output) into a text format

```
seqc -in:profile:asn1=d1or4A_.asn1 -out:profile:txt
```

1.7 As in **recipe 1.5** (i.e. **.asn1** -> **.txt**), but this time reorder profile columns

```
seqc -in:profile:asn1=d1or4A_.asn1 -out:profile:txt  \
    -out:profile:columns=GAPVILMCHWFYKRQDNQST
```

1.8 Sort sequences from the longest to the shortest

```
seqc -in:fasta=in.fasta -seqc:sort -out:fasta
```

This recipe can obviously be combined with the one above (every FASTA sequence in a single line)

1.9 Basic sequence filtering

```
seqc -in:fasta=in.fasta -seqc:sort -select::sequence::protein -out:fasta \
    -select::sequence::long_at_least=30
```

Print only amino acid sequences (due to `-select::sequence::protein` filter) that are at least 30 residues long

1.10 Basic sequence filtering: keep nucleotide sequences

```
seqc -in:fasta=in.fasta -seqc:sort -select::sequence::nucleic -out:fasta \
    -select::sequence::long_at_least=30
```

Print only nucleic acid sequences (due to `-select::sequence::nucleic` filter) that are at least 30 residues long

4.2.3 2. `strc` recipes

2.1 Write only chain A of the given input PDB file

```
strc -in:pdb=5edw.pdb -select::chains=A -out:pdb=5edwA.pdb
```

2.2 Write only aminoacids of chain A (ligands, water etc will be removed)

```
strc -in:pdb=5edw.pdb -select::chains=A -out:pdb=5edwA.pdb -select::aa
```

2.3 Write only selected fragment of a given protein (residues from 1 to 83 of chain A)

```
strc -in:pdb=1PQX.pdb -select::substructure=A:1-83 -op=out.pdb
```

4.2.4 3. `str_calc` recipes

3.1 Find all pairwise all-atom crmsd distances between all the models in a given PDB

```
str_calc -in:pdb=2kmk-1.pdb -calc::crmsd -in:pdb::all_models -
↪in:pdb:native=2KMK.pdb.gz
```

3.2 Read in only CA atoms; find all pairwise crmsd distances between all the models in a given PDB

```
str_calc -select::ca -in:pdb=2kmk-1.pdb -calc::crmsd -in:pdb::all_models \
        -in:pdb:native=2KMK.pdb.gz
```

3.3 Generate *theoretical* NOE restraints on for a protein backbone

```
str_calc -in::pdb=2kwi.pdb -in:pdb:with_hydrogens \
  -calc::distmap::describe -calc::distmap::allatom
```

This command lists all distances between any two backbone atoms; `-in:pdb:with_hydrogens` option forces BioShell to read hydrogen atoms, which is false by default, `-calc::distmap::describe` turns on longer atom descriptions. The output may look as below:

```
A GLN    9  N     10  A GLY    8  N      1     3.602
A GLN    9  N     10  A GLY    8  CA     2     2.418
A GLN    9  N     10  A GLY    8  C      3     1.326
A GLN    9  N     10  A GLY    8  O      4     2.245
A GLN    9  N     10  A GLY    8  HA2    8     2.506
A GLN    9  N     10  A GLY    8  HA3    9     2.959
A GLN    9  CA    11  A GLY    8  N      1     4.834
A GLN    9  CA    11  A GLY    8  CA     2     3.788
A GLN    9  CA    11  A GLY    8  C      3     2.425
A GLN    9  CA    11  A GLY    8  O      4     2.756
```

```
str_calc -in::pdb=2kwi.pdb -in:pdb:with_hydrogens -calc::distmap::describe \
    -calc::distmap::allatom  | awk '{if(($11<2.5) && ($3-$8>4)) print $0}'
```

This output is the filtered with awk. The ouput lines must satisfy the criteria: distance below 2.5 Angstroms, sequence separation at least 4 residues.

3.3 Find all-atom crmsd distances between all models in a single PDB and the reference native structure

```
str_calc -in:pdb=2kmk-1.pdb -calc::crmsd -in:pdb::all_models -
↪in:pdb:native=2KMK.pdb.gz
```

3.4 As in the above example, but after superimposing alpha-carbons, calculate crmsd on all the atoms:

```
str_calc -in:pdb=2kmk-1.pdb -calc::crmsd -in:pdb::all_models -
↪in:pdb:native=2KMK.pdb.gz \
        -calc::crmsd::matching_atoms=A:*:_CA_ -calc::crmsd::rotated_
↪atoms=A:*:*
```

Check peptide docking results: superimpose two structures using alpha carbons of chain A (i.e. the receptor) and calculate crmsd of CA atoms of chain B (i.e. the ligand)

```
str_calc -in:pdb=model-1.pdb -calc::crmsd -in:pdb::all_models -
↪in:pdb:native=native.pdb \
        -calc::crmsd::matching_atoms=A:*:_CA_ -calc::crmsd::rotated_
↪atoms=B:*:_CA_
```

3.5 Check peptide docking results: superimpose two structures using alpha carbons of chain A (i.e. the receptor) and calculate crmsd of CA atoms of chain B (i.e. the ligand)

```
str_calc -in:pdb=models-1.pdb -calc::crmsd -in:pdb::all_models -
↪in:pdb:native=native.pdb \
        -calc::crmsd::matching_atoms=A:*:_CA_ -calc::crmsd::rotated_
↪atoms=B:*:_CA_
```

Note, that this recipe loads **all models** from the `models-1.pdb` file. For instance, if that file contains 10 structures, one can expect the following output:

```
#  name    crmsd  len  crmsd  len
models-1-1.pdb  0.000   96  0.000    4
models-1-2.pdb  0.262   96 22.598    4
models-1-3.pdb  0.274   96 16.670    4
models-1-4.pdb  0.260   96 16.123    4
models-1-5.pdb  0.292   96 24.524    4
models-1-6.pdb  0.320   96 27.575    4
models-1-7.pdb  0.351   96 24.200    4
models-1-8.pdb  0.385   96 24.613    4
models-1-9.pdb  0.297   96 22.778    4
models-1-10.pdb  0.325   96 25.136    4
```

The first column identifies a model structure (name-of-input-file + dash + model number), the second and third provide crmsd on the atoms used for superposition (CA atoms of chains A inthis case) and the number of these atoms (here 96), respectively. Finaly the last two columns provude crmds and atom count for the rotated atom set. The results come for tetrapeptide docking experiment, hence only 4 CA atoms were rotated.

4.2.5 4. `clust` recipes

4.1 Calculate hierarchical clustering of 140 elements; distances are stored in `tm_dist` file.

```
clust -i=tm_dist -n=140 -clustering:out:distance=0.4
```

Prints clusters for critical distance 0.4. By default *single link* clustering strategy is used

4.2.6 5. `hist` recipes

5.1 Calculate a histogram from the 14th column of a given input file:

```
hist -in:file=default.fsc -in:column=14 -hist:x_max=10 -hist:x_min=0
```

The command reads a score file produced by Rosetta and makes a histogram of crmsd, assuming it's in the 14th column

CHAPTER 5

BioShell examples

The latest BioShell 3.0 distribution provides an extensive set of examples. The purpose to create them is three-fold:

- to facilitate continuous testing of the package (unit and integration tests)
- to provide additional functionality to the package,and
- to serve as coding examples and provide ready-to-use snippets

All the tests, which in practice are small C++ applications, were divided into two broad groups; the tests are named staring from `ap_`, `ex_` and `ww_`.

5.1 *ap_** programs

These are integration tests, that besides testing whether the package is bug-free, should also do something usefull.

5.1.1 ap_BackboneHBondMap

> Reads a PDB file and calculates a map of backbone hydrogen bonds, providing also the geometry of each bond.

The results' table printed on stdout looks like below:

# 42 hydrogen bonds found in backbone: TYR 3 -> THR 18 : 2.620 3.346 165.58 94.25 -1.170 -0.702 0.211 2.515 16.25 163.29 LYS 4 -> LYS 50 : 2.259 3.156 120.71 159.88 -1.310 1.445 1.249 1.205 57.75 40.83 LEU 5 -> THR 16 : 1.838 2.802 143.84 -115.27 -2.834 0.102 -1.075 1.488 35.98 -84.57 . . . .

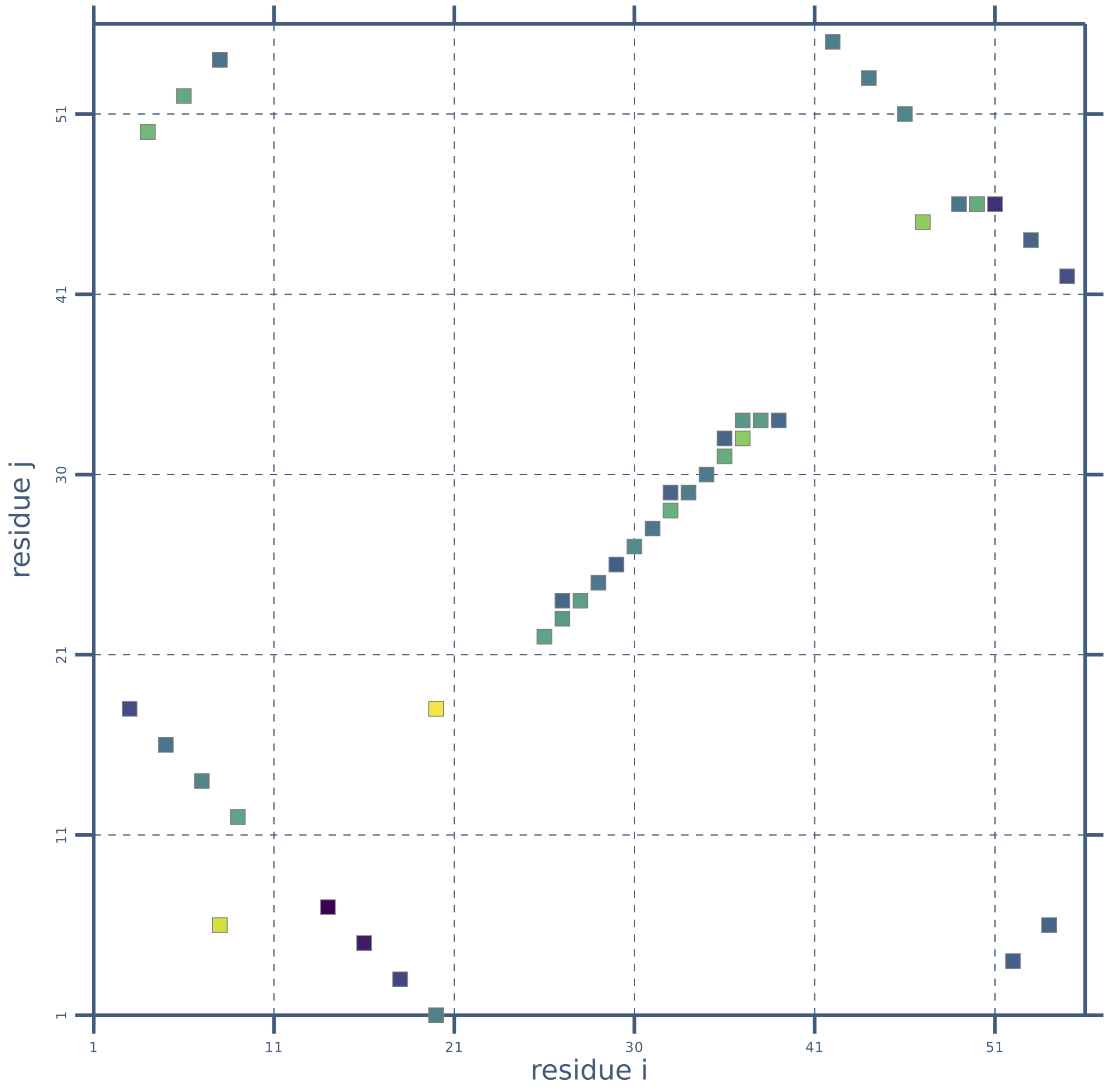
and provides: - H donor residue name and id (columns 1 and 2) - H acceptor residue name and id (columns 4 and 5) - two distances: r(O..H) and r(N..O) (columns 7 and 8) - planar(C-O..H) and dihedral(C-O..H-N) (columns 9 and 10) - DSSP energy for this bond (column 11) - X,Y,Z coordinates of H atom in the local coordinates system (columns 12, 13 and 14) - theta, phi spherical coordinates of H atom (columns 15 and 16) USAGE: ap_BackboneHBondMap 5edw.pdb

Keywords:

- PDB input
- Hydrogen bonds
- data_structures/PairwiseResidueMap
- Protein structure features

Categories:

- core/calc/structural/BackboneHBondMap



5.1.2 ap_Crmsd

ap_Crmsd calculates crmsd value on C-alpha coordinates. The program prints just the crmsd value.

USAGE: ap_Crmsd structureA.pdb [structureB.pdb]

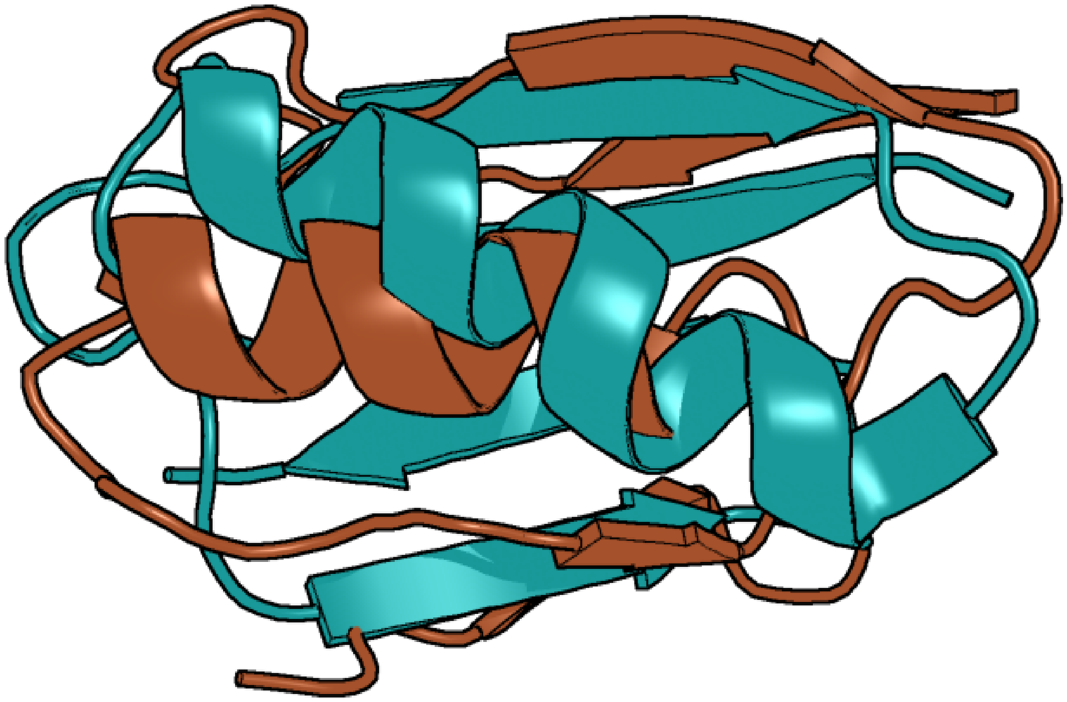
If two structures are provided, the program calculates crmsd between the first model of structure A and the first model of structure B. If only one input PDB file is given, crsmd is computed for every pair of models found in the input file (each-vs-each)

Keywords:

- PDB input
- crmsd

Categories:

- core/calc/structural/transformations/Crmsd



5.1.3 ap_Hexbins

Reads a file with 2D observations (two columns with real values) and makes hexbin histogram.

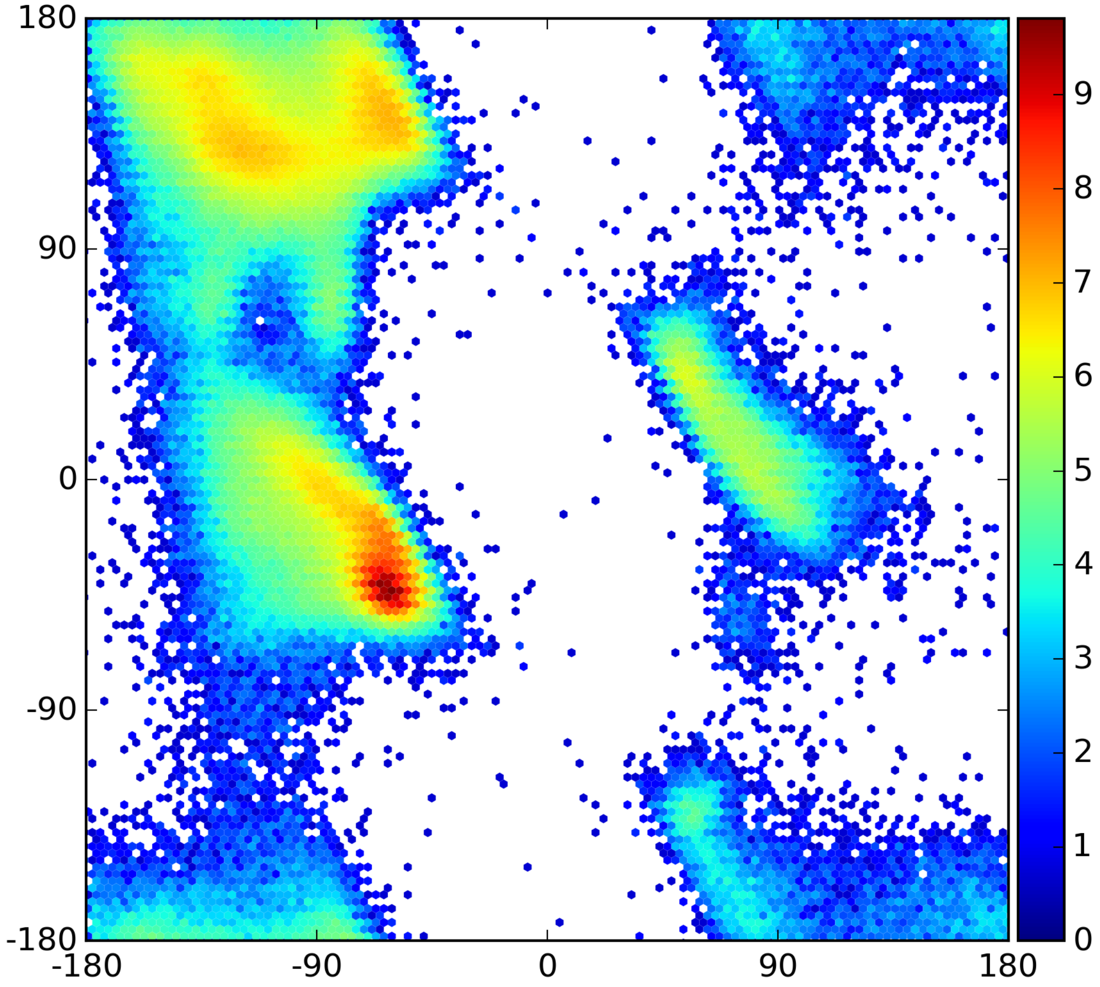
USAGE: ap_Hexbins input.dat [bin_side_width]

Keywords:

- histogram 2D
- plotting

Categories:

- core::calc::statistics::Hexbins



5.1.4 ap_aligned_pdb

Reads an alignment between two proteins (PIR format) and the two respective protein structures (PDB format)

and writes the aligned parts of the two structures.

The program concerns only the first two sequences found in the PIR file; they must be given in the same order as the input PDB files. Only the first chain will be used from either structure; if you need to superimpose chain 'B', use strc command to extract it prior using ap_aligned_pdb.

The program writes 'query' and 'tmplt' files which contain the respective structure fragments, already superimposed (the template on the query). One of the two structures may be missing (either the query or the template), dash '-' should be used instead of the respective file name, as in the examples below.

USAGE: ap_aligned_pdb example.pir prot1.pdb prot2.pdb ap_aligned_pdb example.pir - prot2.pdb ap_aligned_pdb example.pir prot1.pdb -

Keywords:

- PDB input
- PIR input
- PDB output

Categories:

- core/data/io/pir_io

5.1.5 ap_chi1_rotamers_estimation

ap_chi1_rotamers_estimation reads a text file with Chi_1 angles (single column of real values)

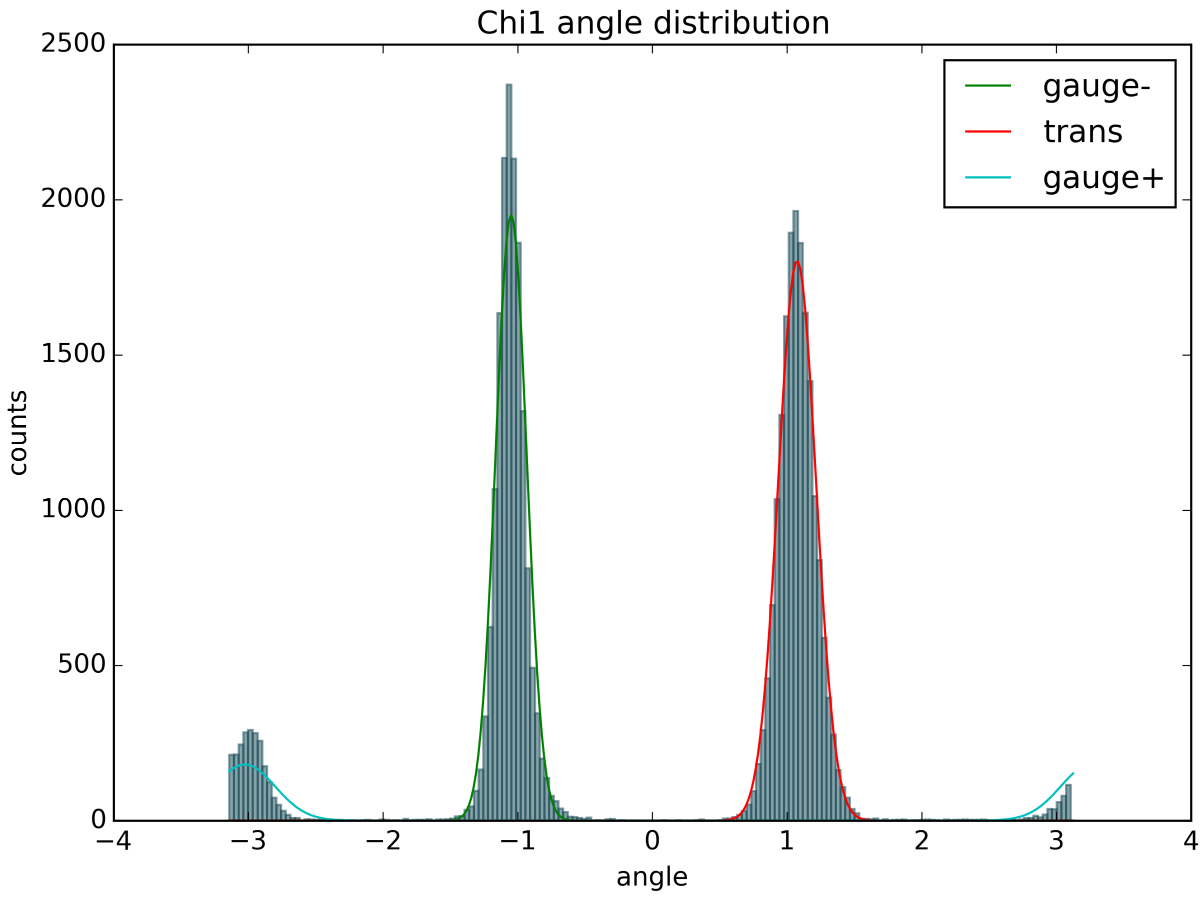
and fits a mixture of VonMisses distributions to the data. The program may be thus used for deriving rotamer library for VAL, THR, SER and CYS USAGE: ap_chi1_rotamers_estimation THR_chi1.dat

Keywords:

- von Misses distribution
- estimation
- expectation-maximization
- statistics

Categories:

- core::calc::statistics::VonMissesDistribution



5.1.6 ap_contact_map

ap_contact_map calculates a contact map for a given protein structure

If a multi-model PDB file was given, the program prints contact count observed in all models

USAGE: ap_contact_map CA 2kwi.pdb.pdb 4.5

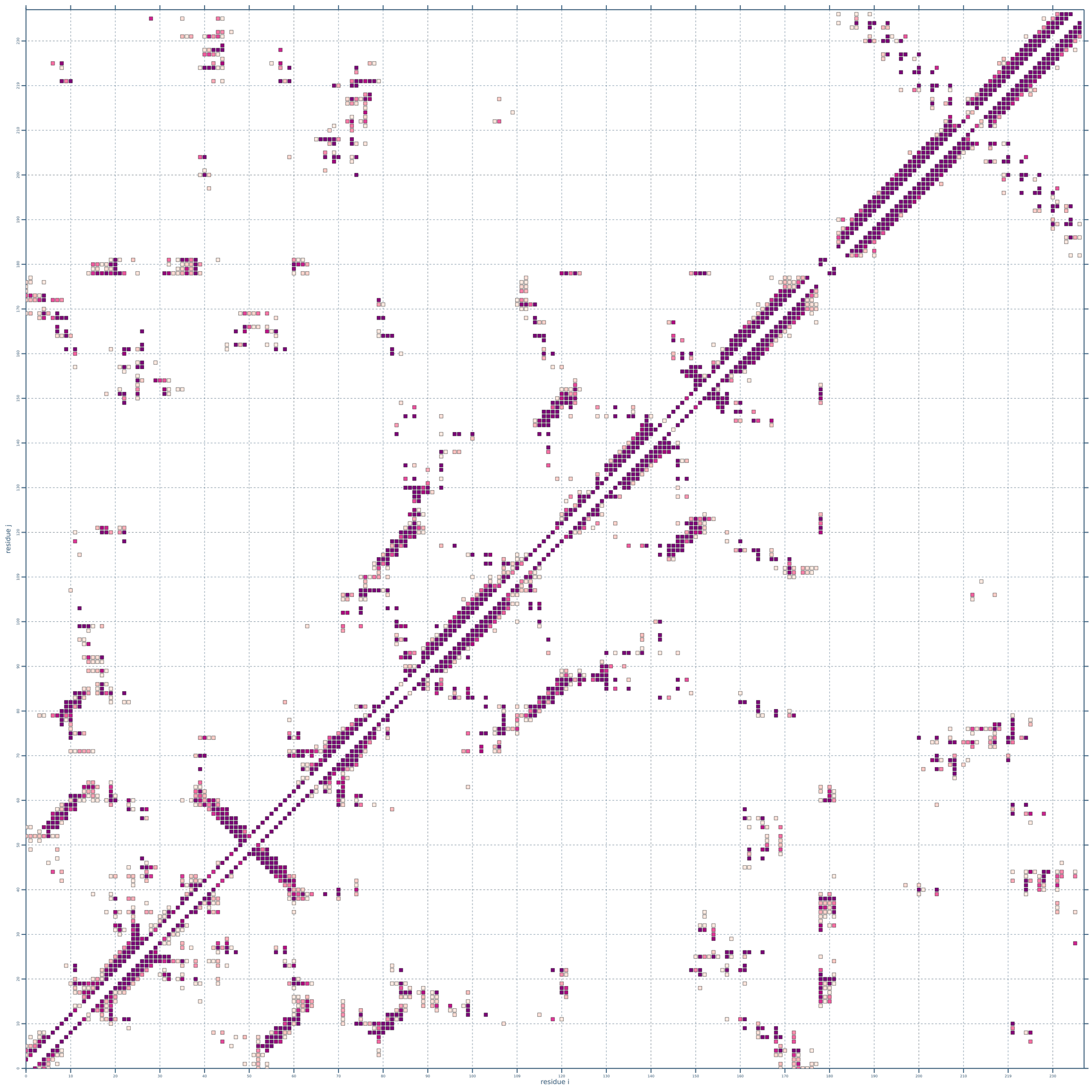
where 2kwi.pdb is the input file and 4.5 the contact distance in Angstroms. CA defines the contact map type; allowed options are: CA CB and SC for Calpha, C-beta and all atom side chain, respectively

Keywords:

- PDB input
- contact map

Categories:

- core::calc::structural::ContactMap



5.1.7 ap_fit_VonMises_mixture

ap_fit_VonMisses_mixture reads a text file with 1D arbitrary observations in degrees

and fits a mixture of VonMisses distributions to the data. The number of distributions to fit is determined by the starting parameters: f$muf$ and f$kappaf$ for each distribution USAGE: ap_fit_VonMises_mixture chi_angles.dat -1.05 30 -3.0 30 1.05 30 ap_fit_VonMises_mixture chi_angles.dat 3

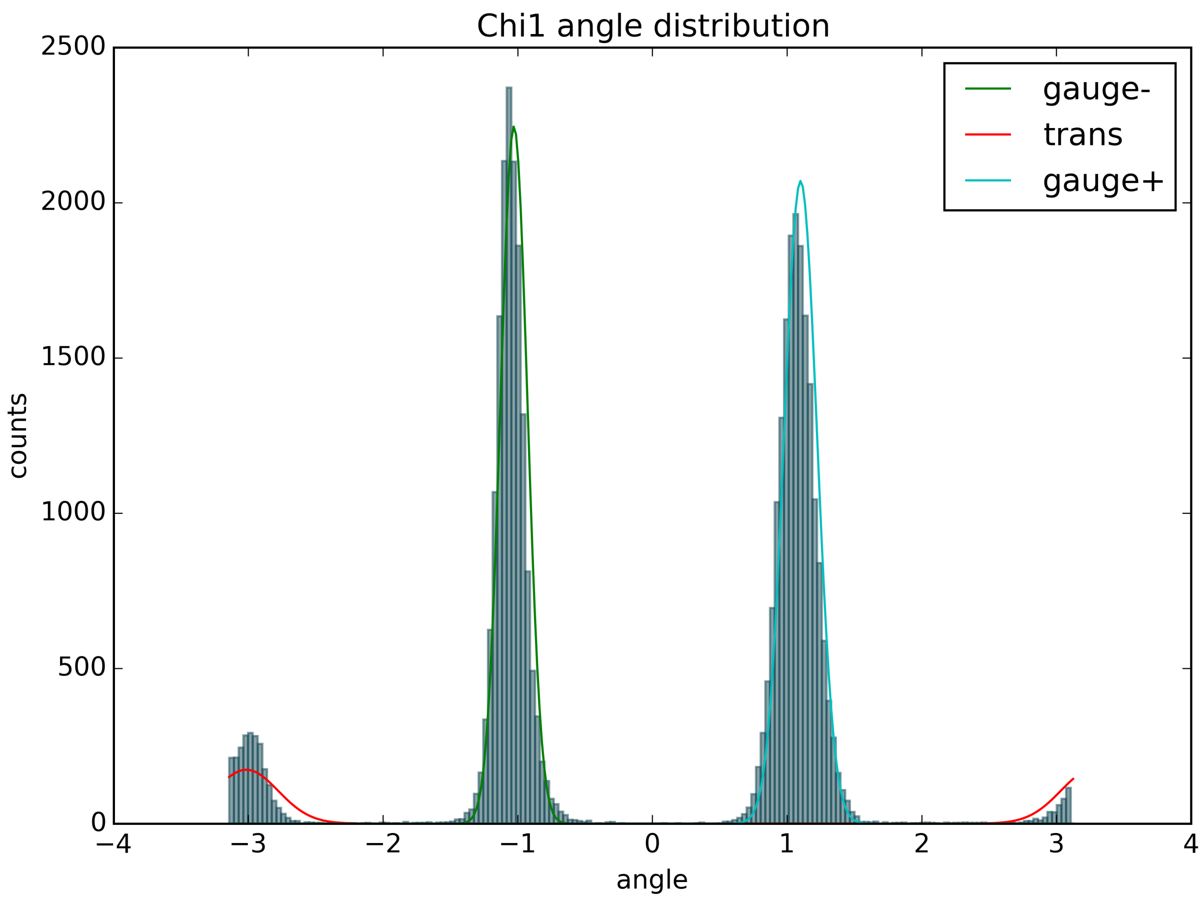
Keywords:

- von Misses distribution
- estimation

- expectation-maximization

Categories:

- core::calc::statistics::VonMissesDistribution



5.1.8 ap_ligand_contacts

ap_ligand_contacts finds contacts between a ligand molecule and a protein.

It reads a multi-model PDB file and detects contacts in everyone of them. The output provides the interacting residues (name and residueId) along with the number of observations for this contact

USAGE: ap_ligand_contacts 5edw.pdb TTP 7.0

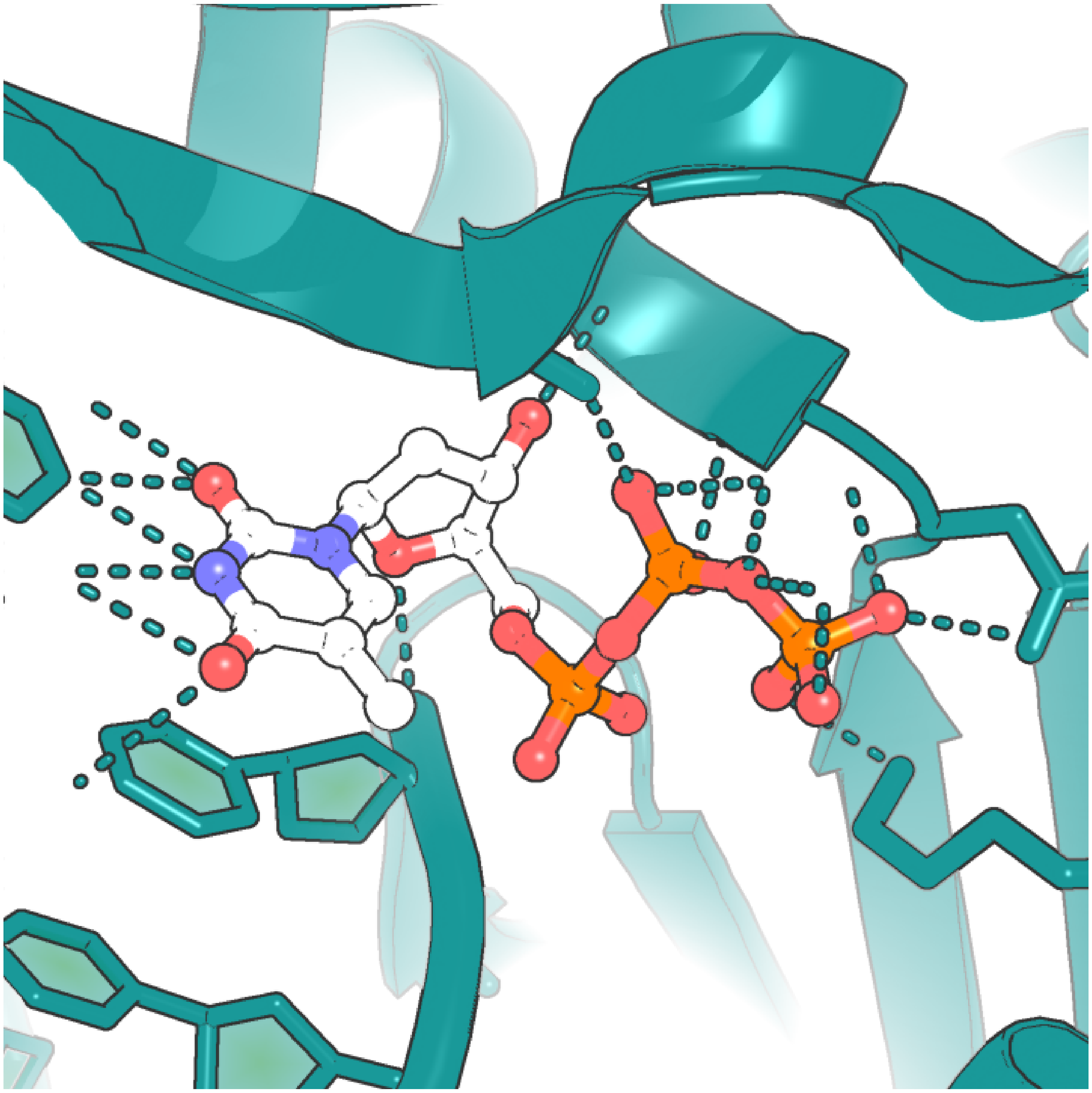
where 5edw.pdb id an input file, TTP the ligand code and 7.0 - contact distance in Angstroms

Keywords:

- PDB input
- contact map
- ligand

Categories:

- core::data::io::Pdb



5.1.9 ap_orient_pdb

ap_orient_pdb reads a PDB file and orients the atoms along the axes so the longest protein dimension is along X

and the second longest along Y. Then a second transformation is created to rotate a structure fragment around Z axis by 45 degrees

USAGE: ap_orient_pdb [2kwi.pdb 419 446]

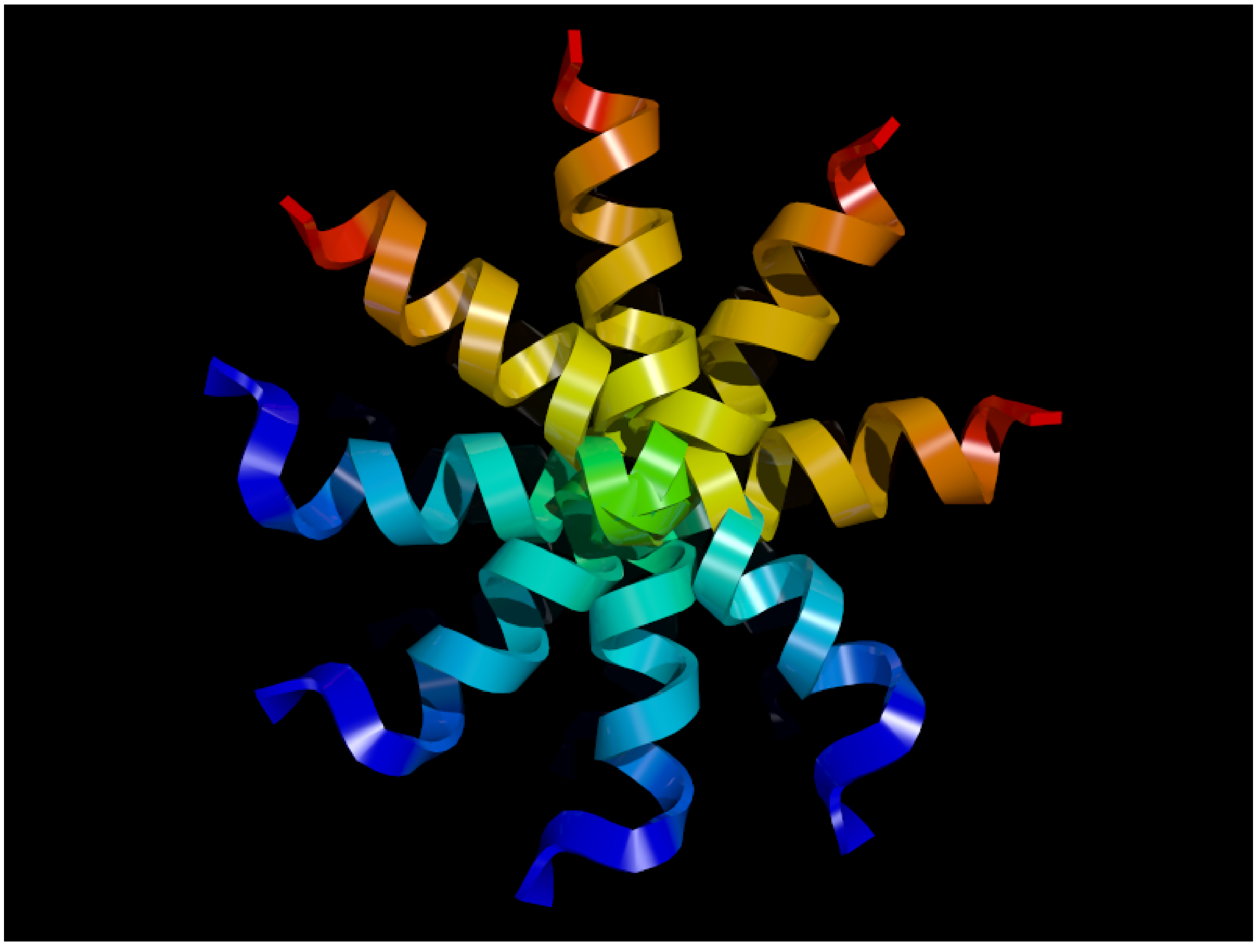
where 2kwi.pdb is the name of an input file and 419 446 is the first and last of the reoriented residues, respectively

Keywords:

- PDB input
- structural fragment
- structure selectors
- PCA
- transformations

Categories:

- core/calc/numeric/PCA.hh



5.1.10 ap_shuffled_sequence_alignment

Reads a FASTA file with two sequences and calculate global sequence alignment scores with one sequence randomly shuffled. The statistics of scores from randomised alignments is then used to estimate p-value of the global alignment.

The program prints all the randomized alignment scores and estimated p-value of the alignment

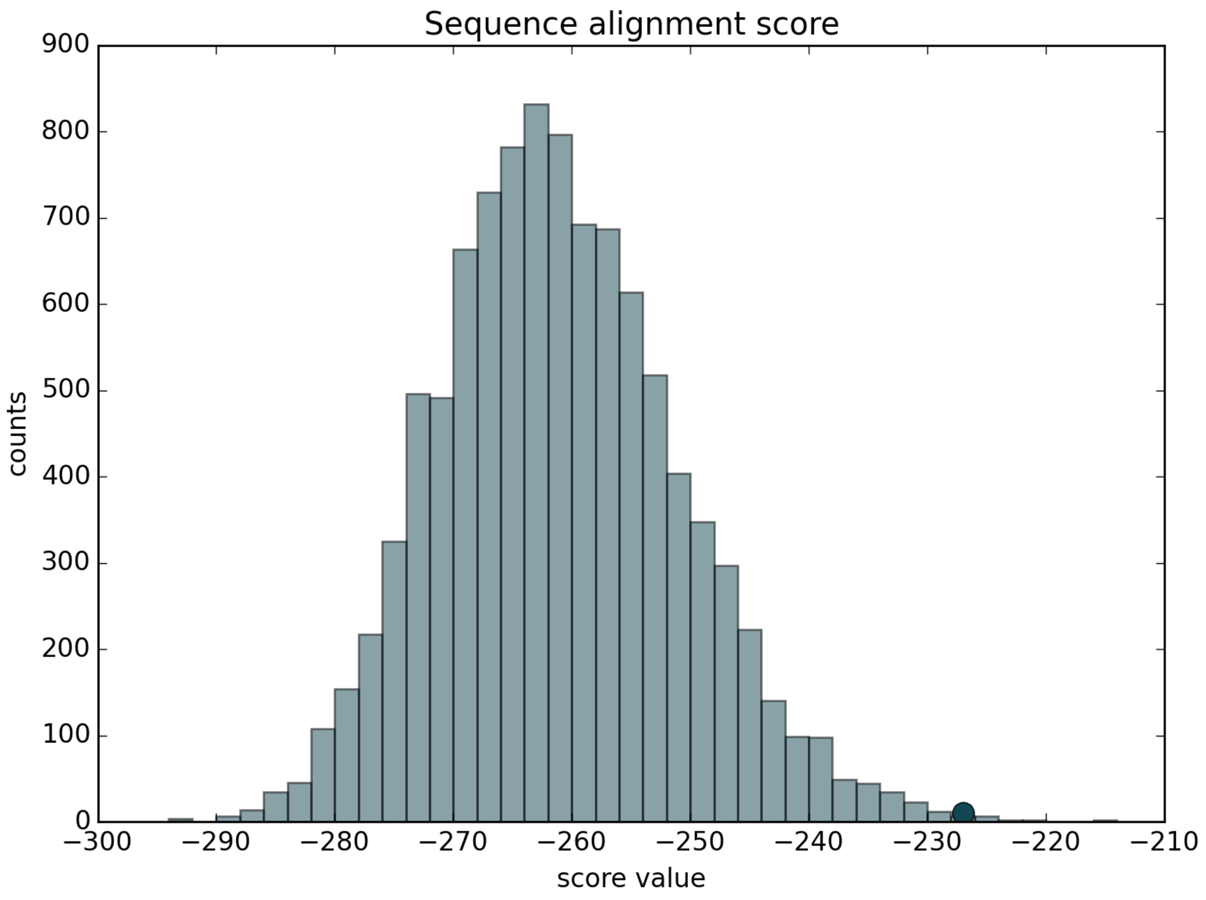
USAGE: ap_shuffled_sequence_alignment test_inputs/ferrodoxins.fasta [N_shuffles]

Keywords:

- FASTA input
- Needleman-Wunsch
- sequence alignment
- alignment p-value

Categories:

- core::alignment::NWAligner



5.1.11 ap_AAHydrophobicity

Reads a PDB file and substitutes b-factor column with hydrophobicity values according to Kyte-Doolittle scale.

If just a PDB file is given as an input, all b-factors will be replaced by respective KD hydrophobicity values. User can also provide a Multiple Sequence Alignment (MSA) in ClustalO format (.aln); hydrophobicity values will be averaged over a corresponding column of the MSA

USAGE: ap_AAHydrophobicity 2gb1.pdb ap_AAHydrophobicity 2gb1.pdb 2gb1.aln 2GB1

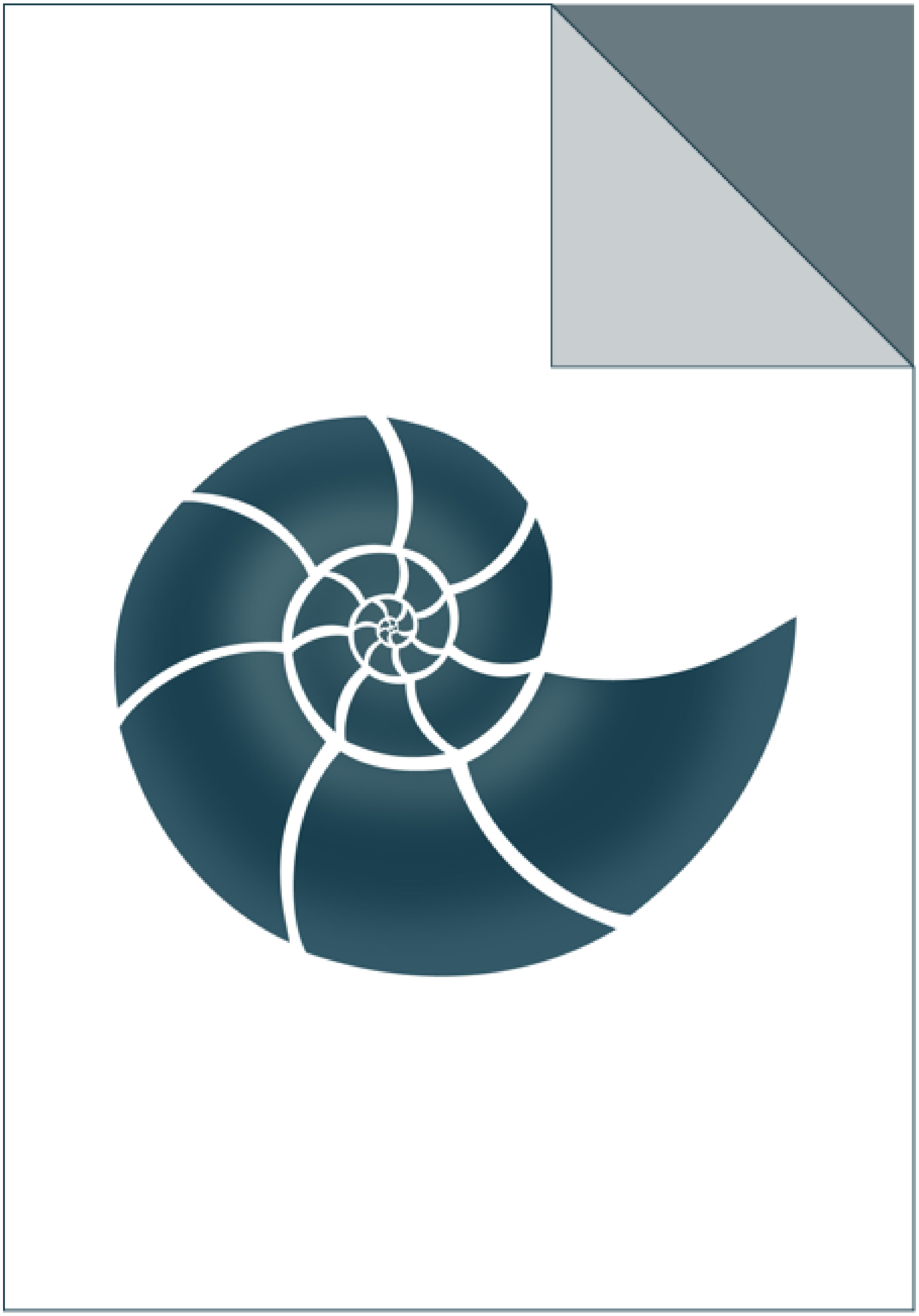
The sequence from the given PDB file must also be included in the alignment; its name is third argument of the program.

Keywords:

- no_keywords

Categories:

- no_categories



5.1.12 ap_AlignmentPValuesProtocol

ap_AlignmentPValuesProtocol evaluates pairwise p-value of sequence alignments between all sequences found in a given FASTA file.

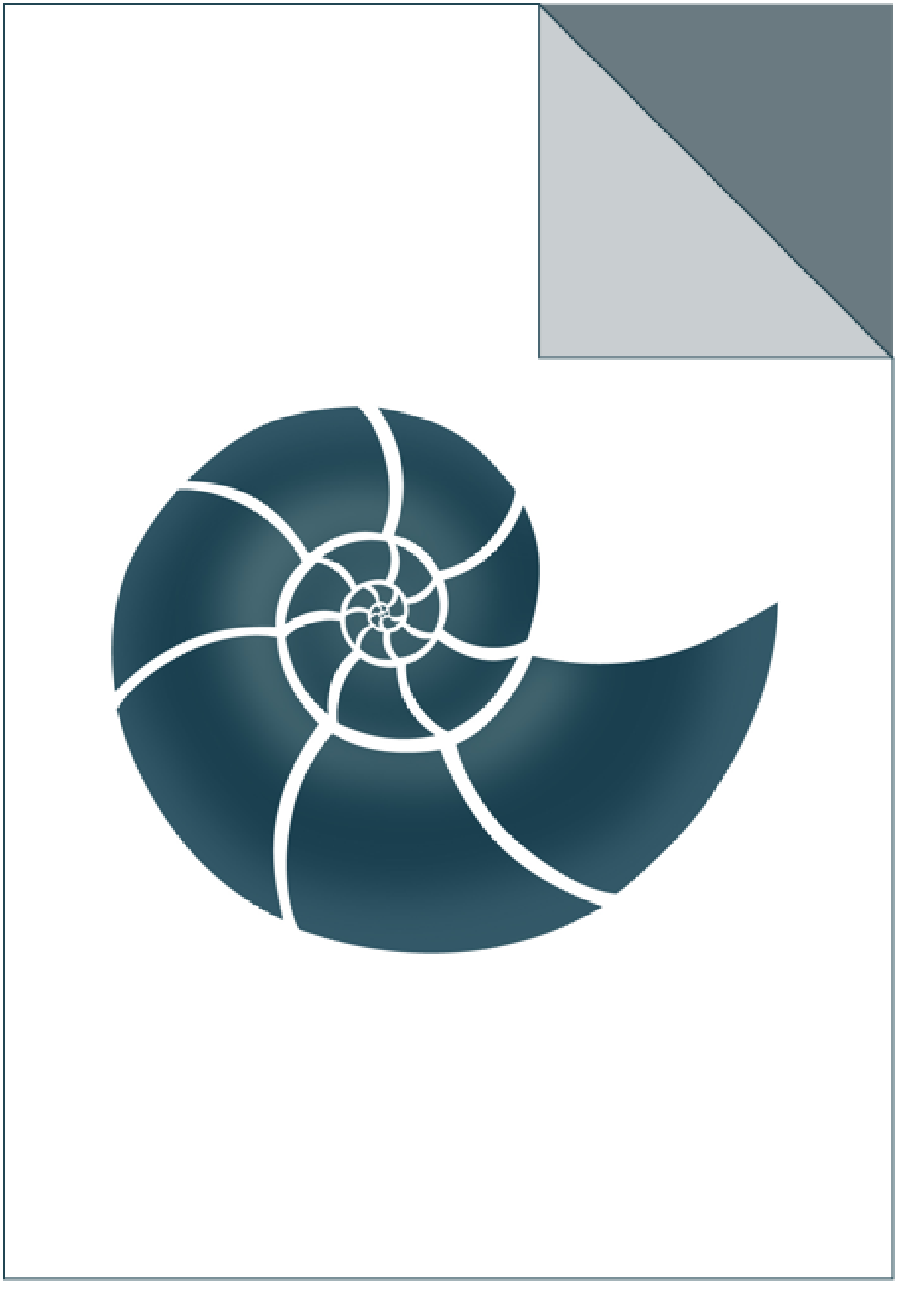
USAGE: ap_AlignmentPValuesProtocol input.fasta

Keywords:

- no_keywords

Categories:

- no_categories



5.1.13 ap_LocalStructureMatch

Finds contiguous structural segments that are similar between two structures.

The program can use only segments of size 7 or 5. It looks for structurally similar segments shared between two PDB files, given as an input.

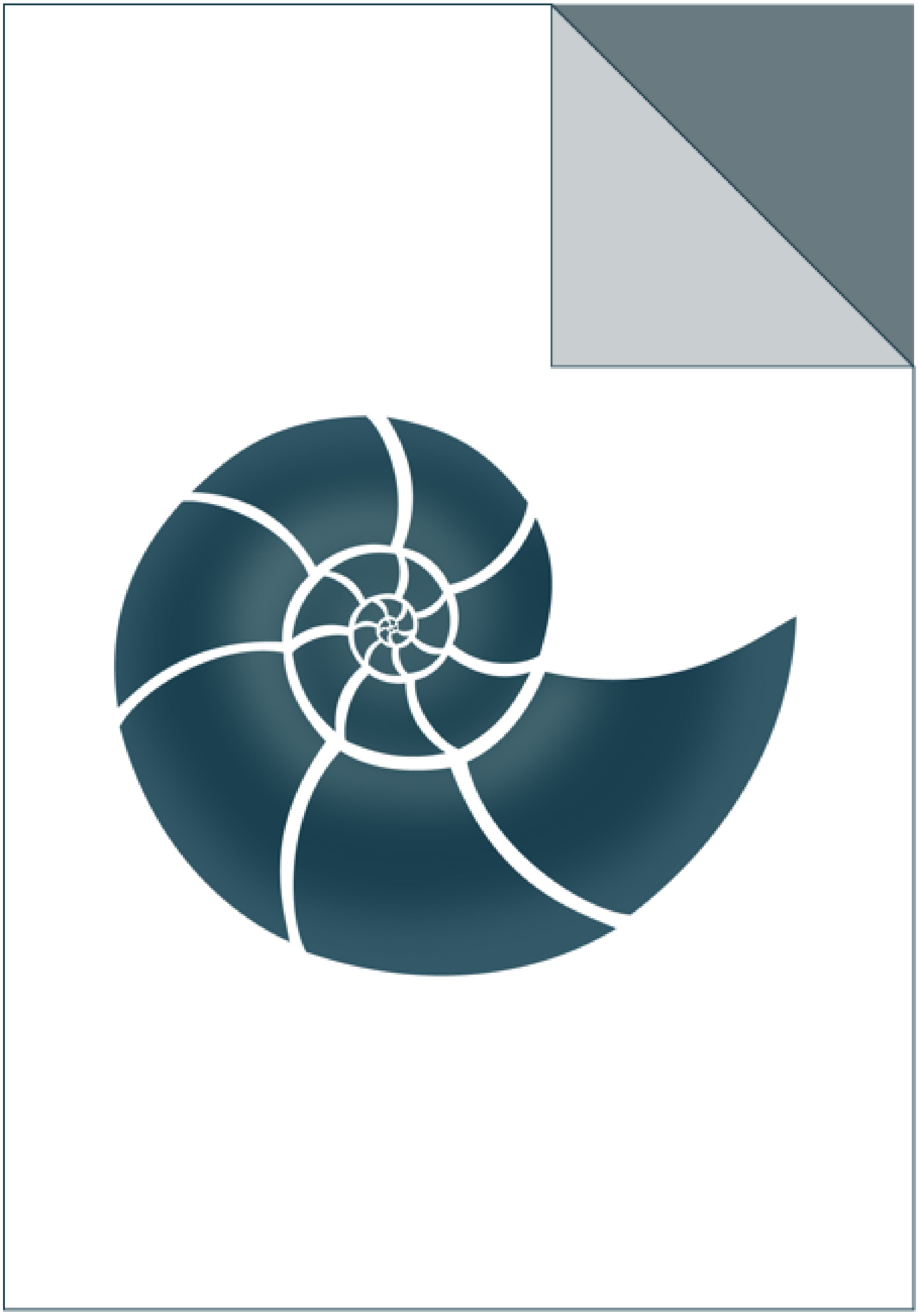
USAGE: ./ap_LocalStructureMatch 7 4rm4A.pdb 5ofqA.pdb

Keywords:

- no_keywords

Categories:

- no_categories



5.1.14 ap_MSAColumnConservation

> ap_MSAColumnConservation reads a Multiple Sequence Alignment (MSA) in ClustalW or in FASTA format

and evaluates sequence conservation for every column

USAGE: ./ap_MSAColumnConservation cyped.CYP109.aln [M5R670_9BACI]

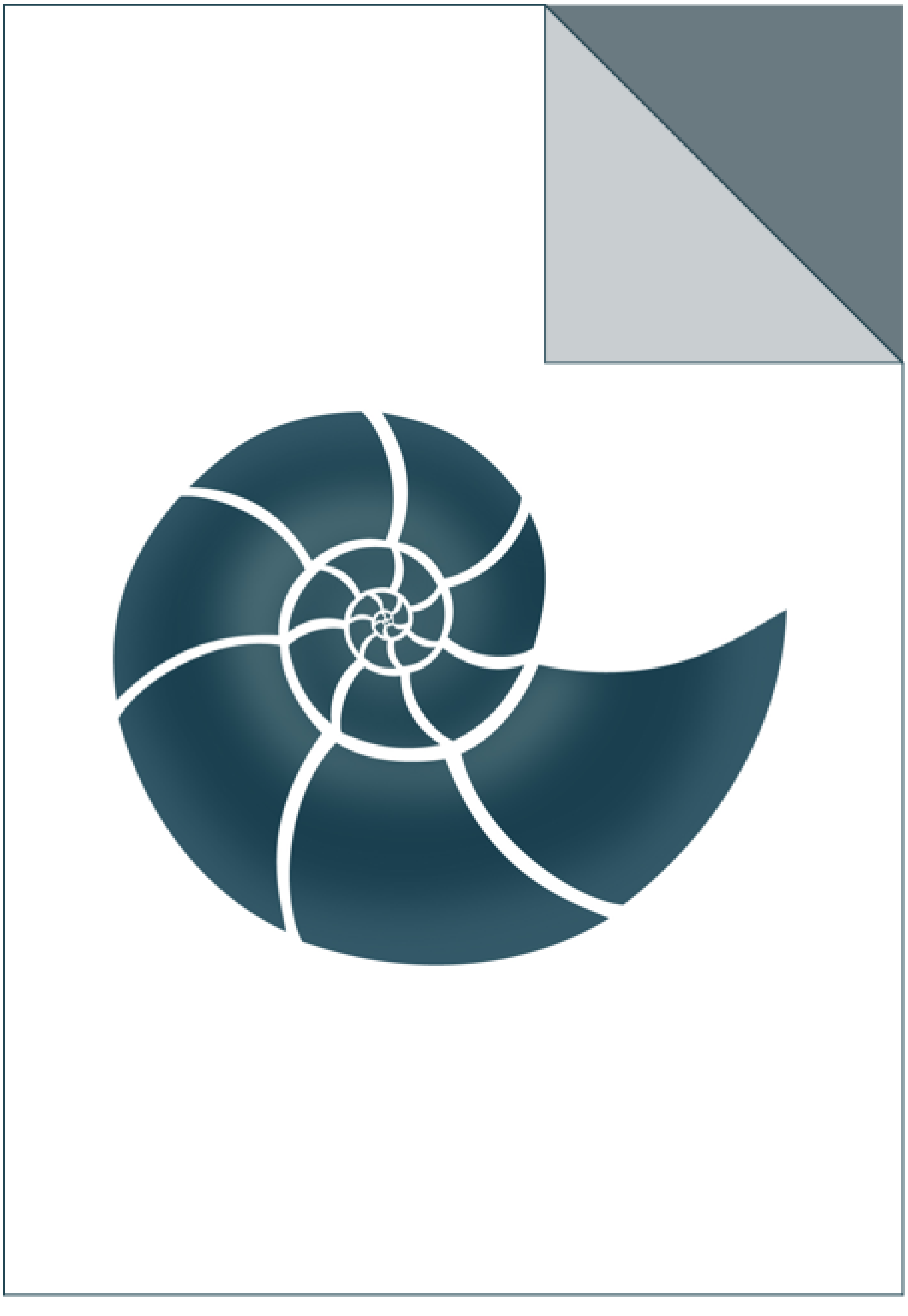
where cyped.CYP109.aln is the name of input MSA file (.aln or .fasta format). If the second optional argument (here: M5R670_9BACI) is given, program will attempt find the sequence annotated by this name. When such a sequence is found, additional column will be added to provide residue for every position in that sequence (gaps are also shown)

Keywords:

- no_keywords

Categories:

- no_categories



5.1.15 ap_NWAligner

Calculate all pairwise global sequence alignments (Needleman & Wunsh algorithm) between sequences read from a FASTA file.

For every pair of sequences it prints three columns: query length, template length and the alignment score.

USAGE: ap_NWAligner test_inputs/ferrodoxins.fasta test_inputs/ferrodoxins.fasta [BLOSUM80]

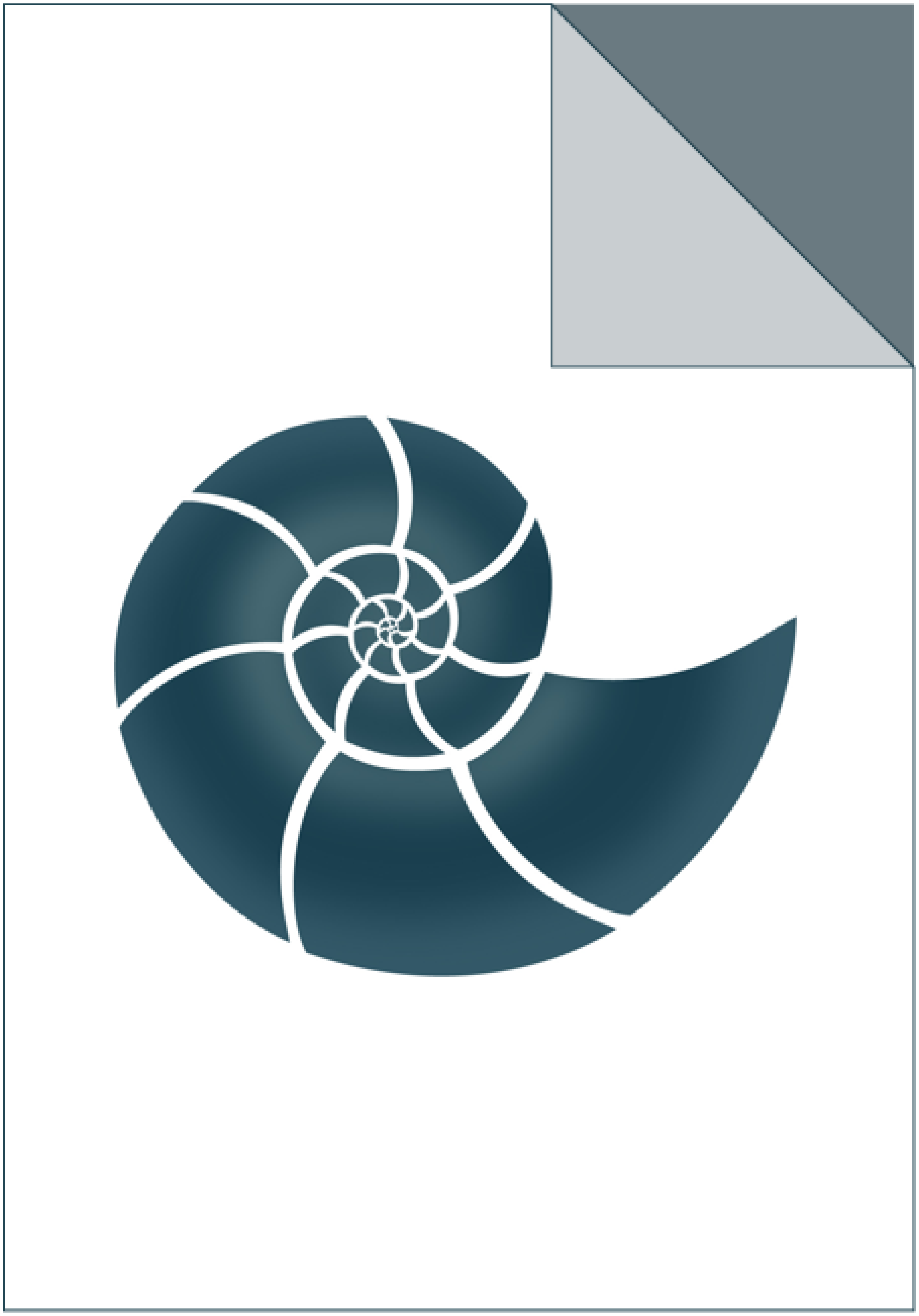
The substitution matrix name (here: BLOSUM80) is an optional parameter.

Keywords:

- no_keywords

Categories:

- no_categories



5.1.16 ap_OnlineStatistics

ap_OnlineStatistics reads a file with real values and calculates simple statistics: min, mean, stdev, max.

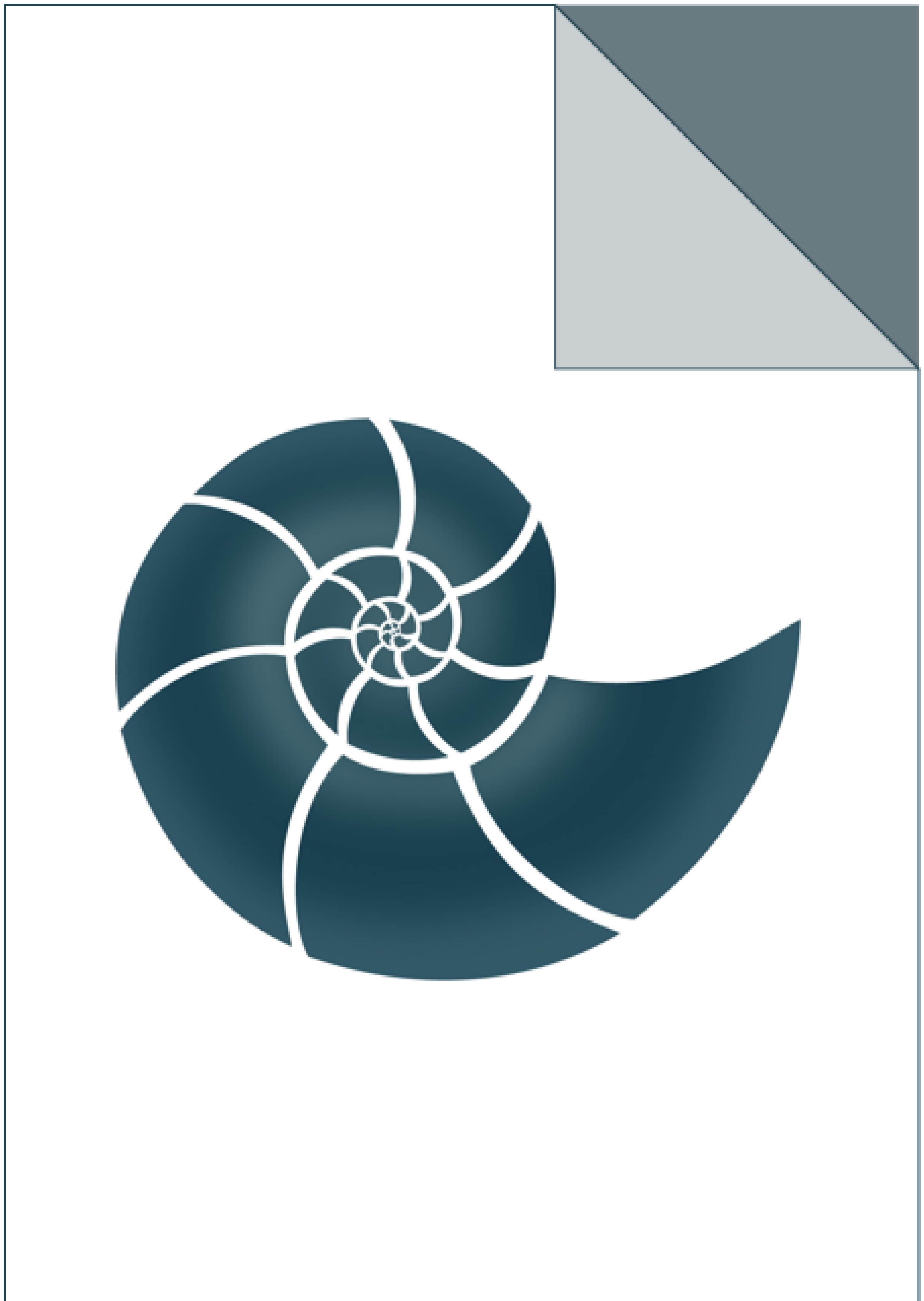
If no input file is provided, the program calculates the statistics from a random sample. USAGE: ap_OnlineStatistics infile

Keywords:

- no_keywords

Categories:

- no_categories



5.1.17 ap_PairwiseCrmsd

ap_PairwiseCrmsd calculates pairwise crmsd values for a set of protein structures (at least two).

This example evaluates crmsd for each pair of proteins twice: on C-alpha atoms and on all backbone atoms

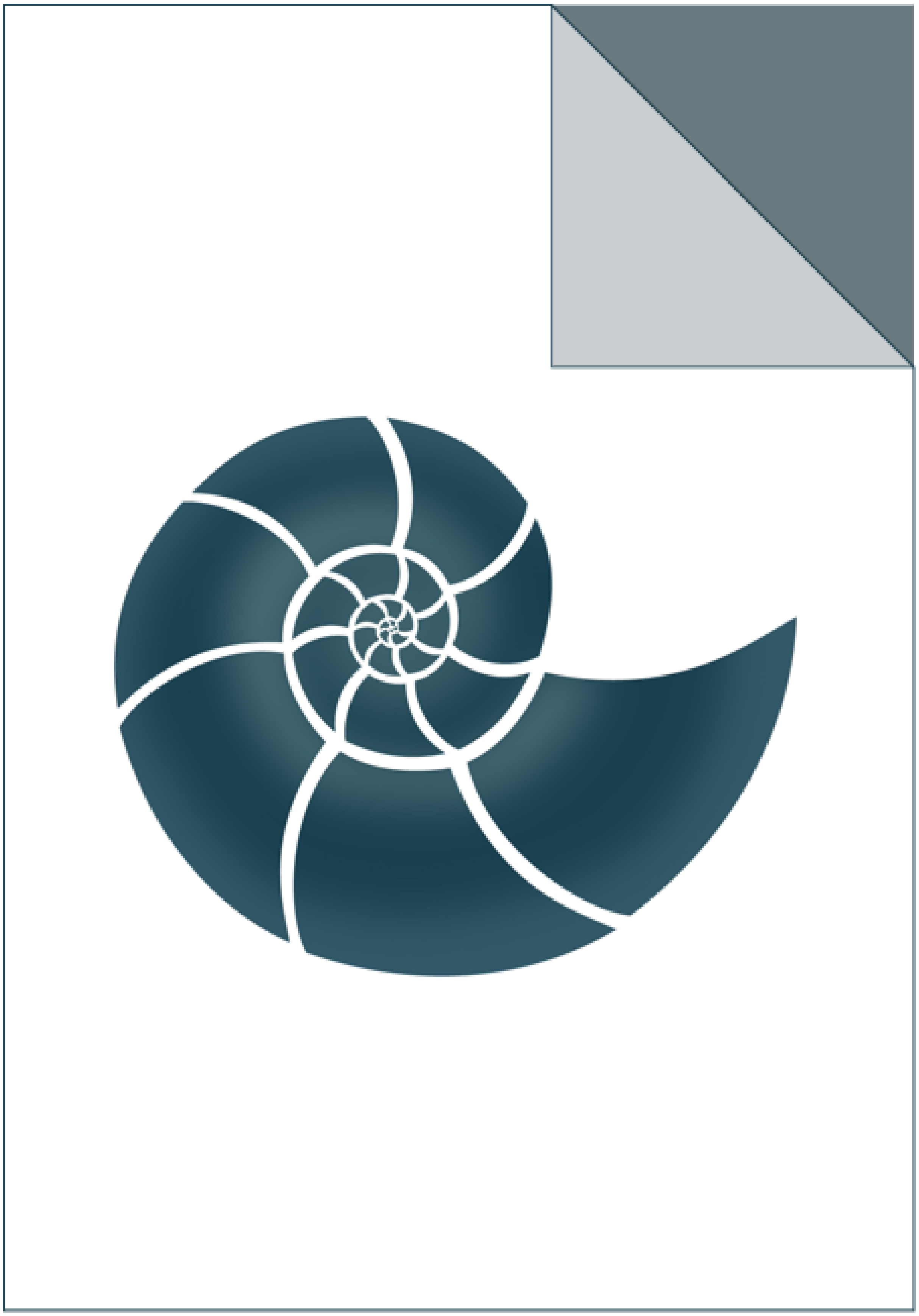
USAGE: ap_PairwiseCrmsd structureA.pdb structureB.pdb [structureC.pdb …]

Keywords:

- no_keywords

Categories:

- no_categories



5.1.18 ap_PairwiseSequenceIdentityProtocol

> ap_PairwiseSequenceIdentityProtocol evaluates pairwise sequence identity between all sequences found in a given FASTA file.

The calculated values are printed on the screen if they are greater than a given cutoff (0.25 in the example below). Calculations may be executed in several parallel threads, the number of threads is the second parameter of this program

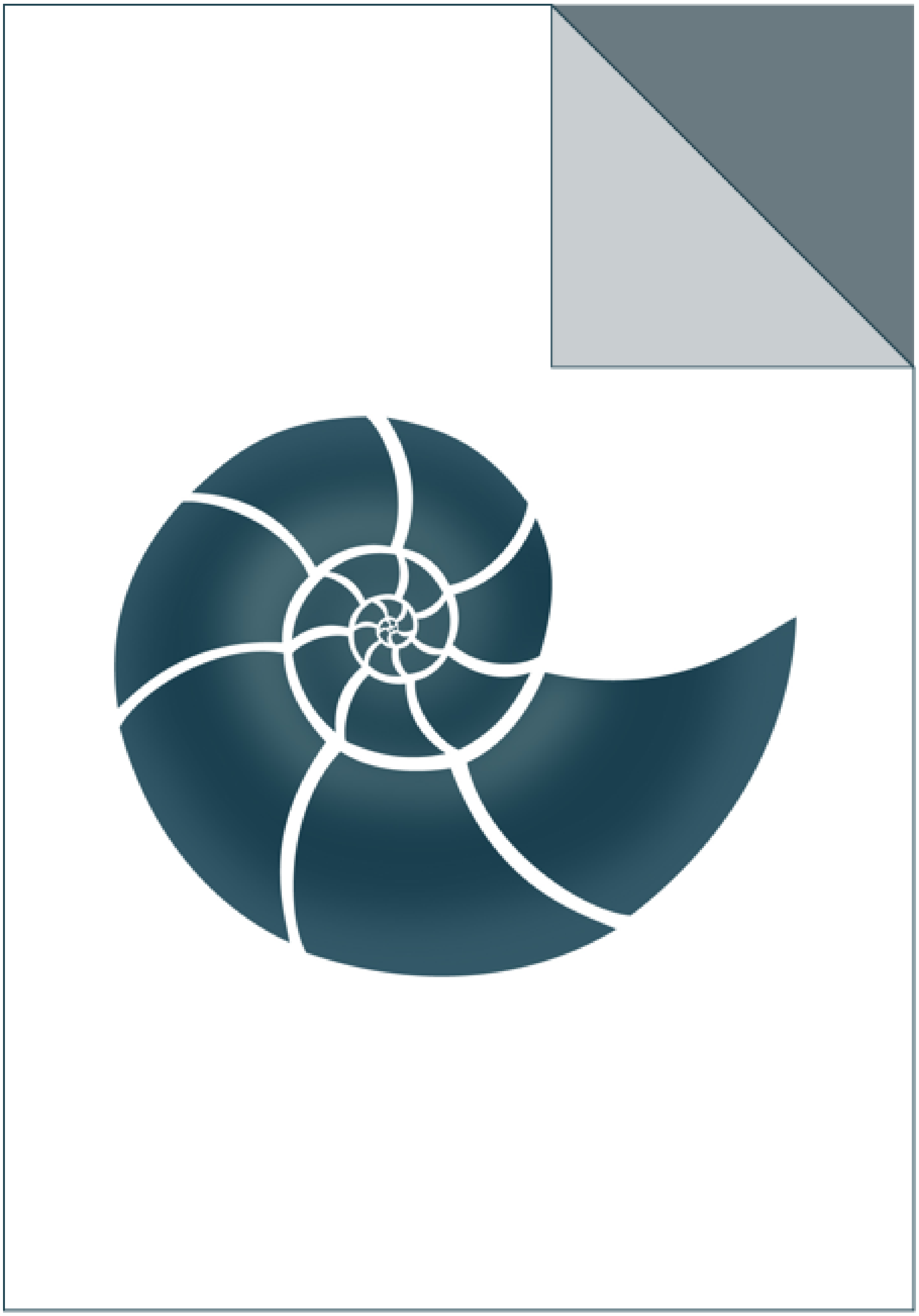
USAGE: ap_PairwiseSequenceIdentityProtocol input.fasta [n_threads [seqID_cutoff [5tuple_cutoff]]] EXAMPLE: ap_PairwiseSequenceIdentityProtocol small500_95identical.fasta 4 0.25

Keywords:

- no_keywords

Categories:

- no_categories



5.1.19 ap_ProteinArchitecture

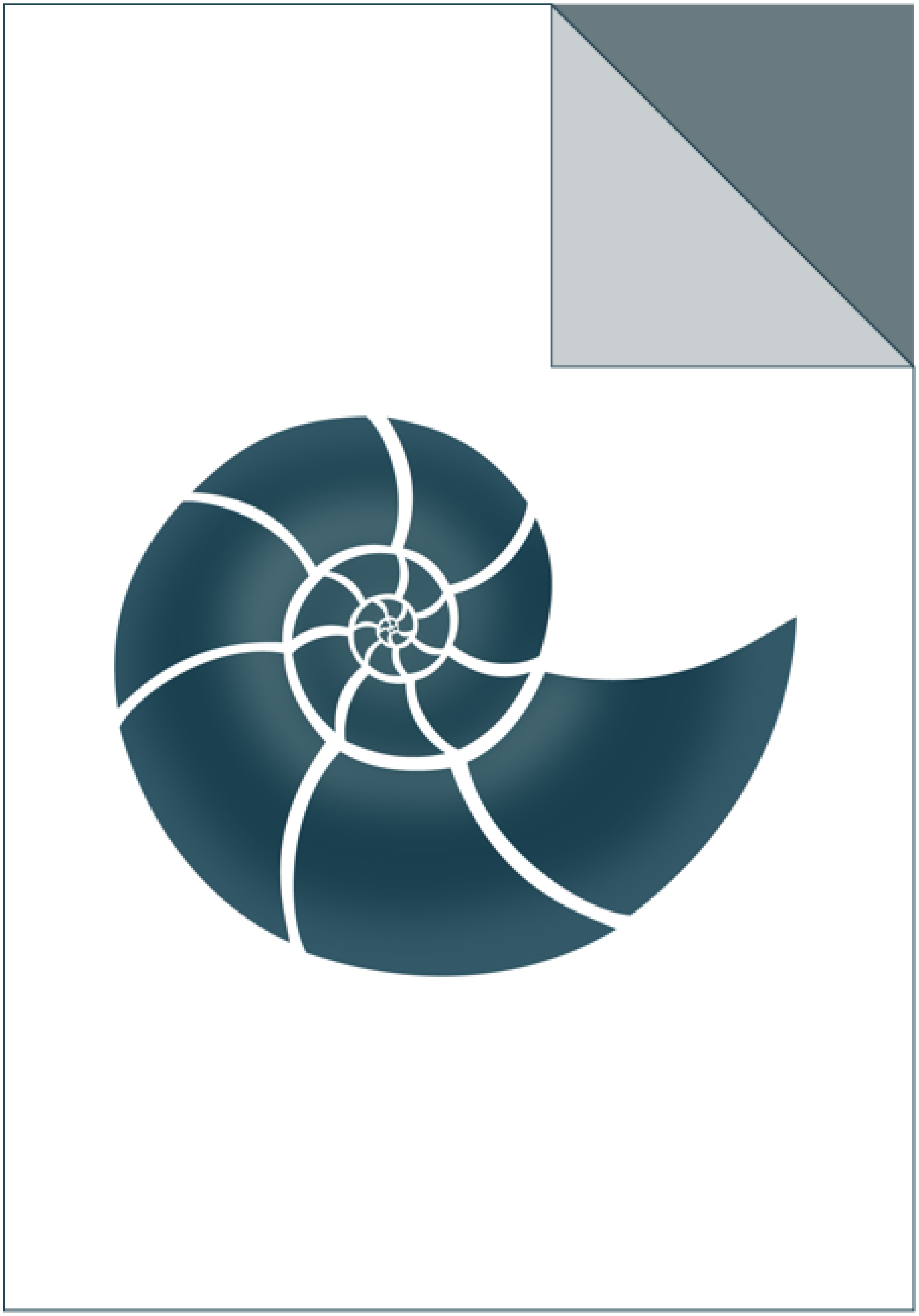
```
ap_ProteinArchitecture 5edw.pdb
```

Keywords:

- no_keywords

Categories:

- no_categories



5.1.20 ap_QuickSequenceIdentity

Estimates pairwise sequence similarity for a set of sequences given in a FASTA format.

The output table has 5 columns: i, j (indexing the pair of aligned sequences), 5-tuple score, 8-tuple score and alignment seq_id, where the first two seq_id values are estimations calculated for 16-aa reduced alphabet and the last value is the true sequence identity evaluated over a sequence alignment.

If the number of sequences in the provided input file does not exceed 1000, true sequence identity is evaluated for all pairs; when the input data set is larger, exact sequence alignment is computed only when 5-tuple score is greater than the given value [score5_to_align, 0.001 by default] and printed when exceeds score5_to_print 0.2 by default. If the 5-tuple score is in between the two cutoff values, true sequence identity will be evaluated and included in a histogram, but not printed. Otherwise the output file might be extremely large.

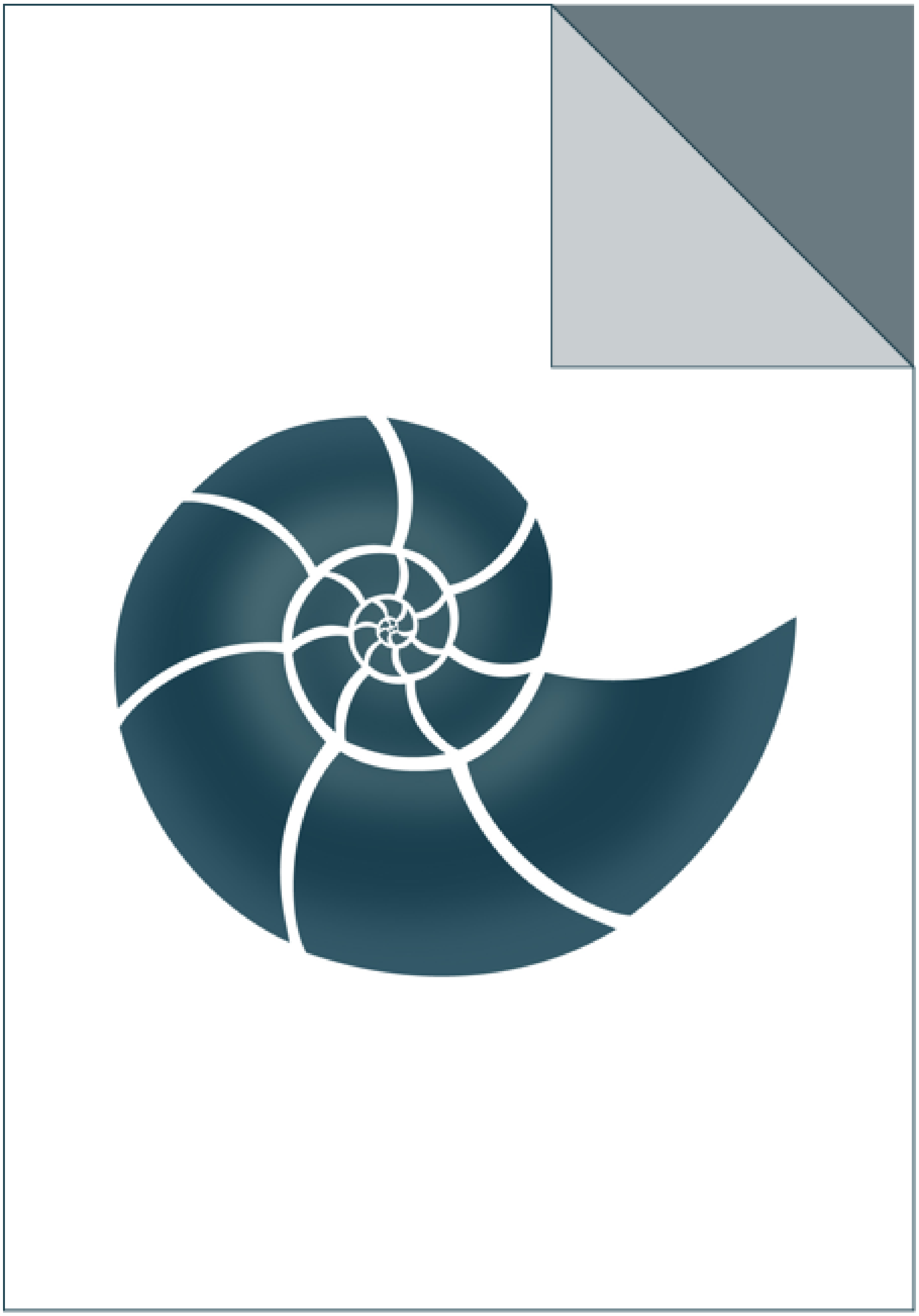
USAGE: ap_QuickSequenceIdentity sequences.fasta [score5_to_align score5_to_print]

Keywords:

- no_keywords

Categories:

- no_categories



5.1.21 ap_SWAligner

Calculate all pairwise local sequence alignments (Smith&Waterman algorithm) between sequences read from a FASTA file.

For every pair of sequences it prints three columns: query length, template length and the alignment score.

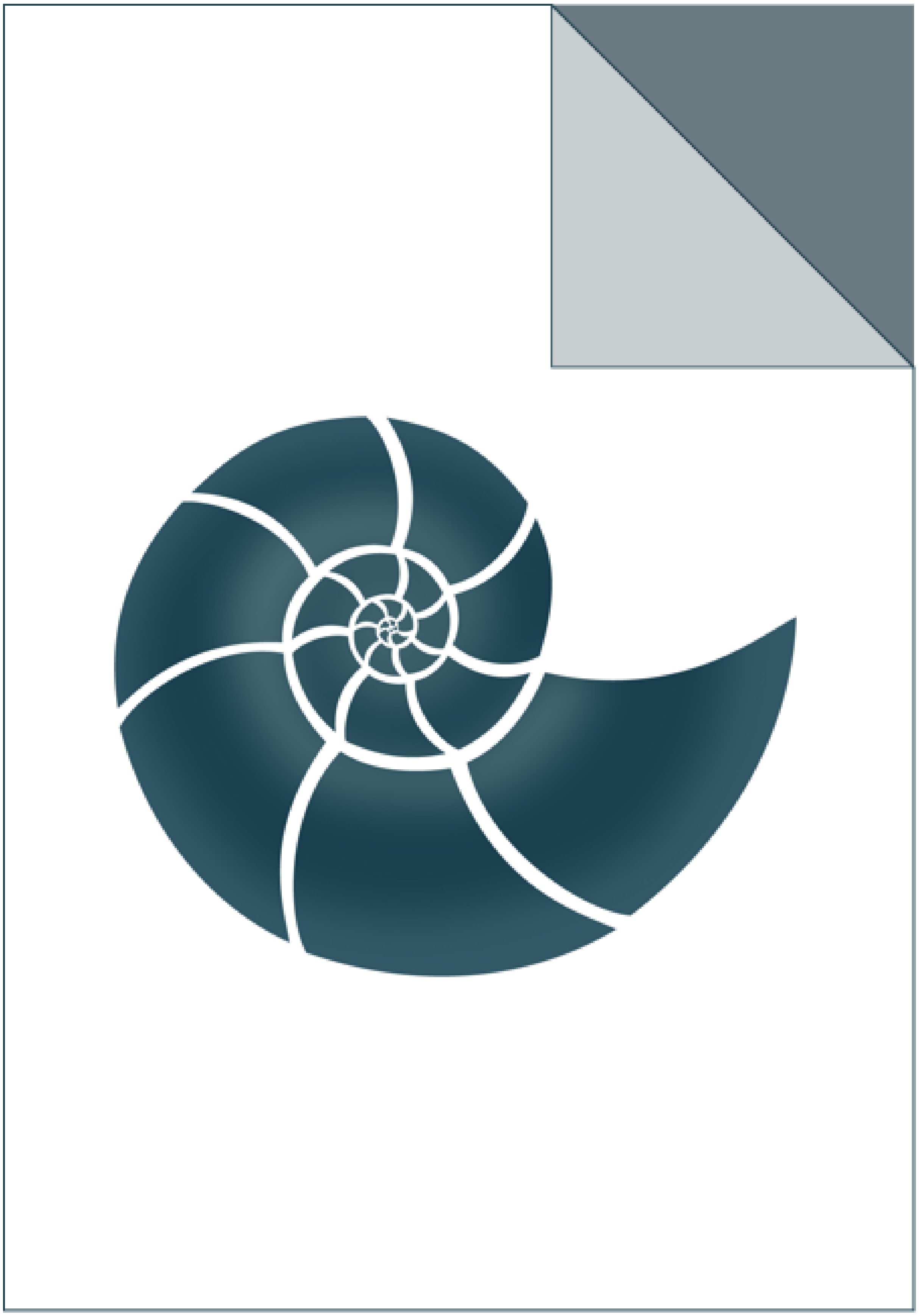
USAGE: ap_SWAligner test_inputs/ferrodoxins.fasta test_inputs/ferrodoxins.fasta

Keywords:

- no_keywords

Categories:

- no_categories



5.1.22 ap_SequenceProfile

ap_SequenceProfile reads a Multiple Sequence Alignment (MSA) in ClustalW format and prints a sequence profile made from it.

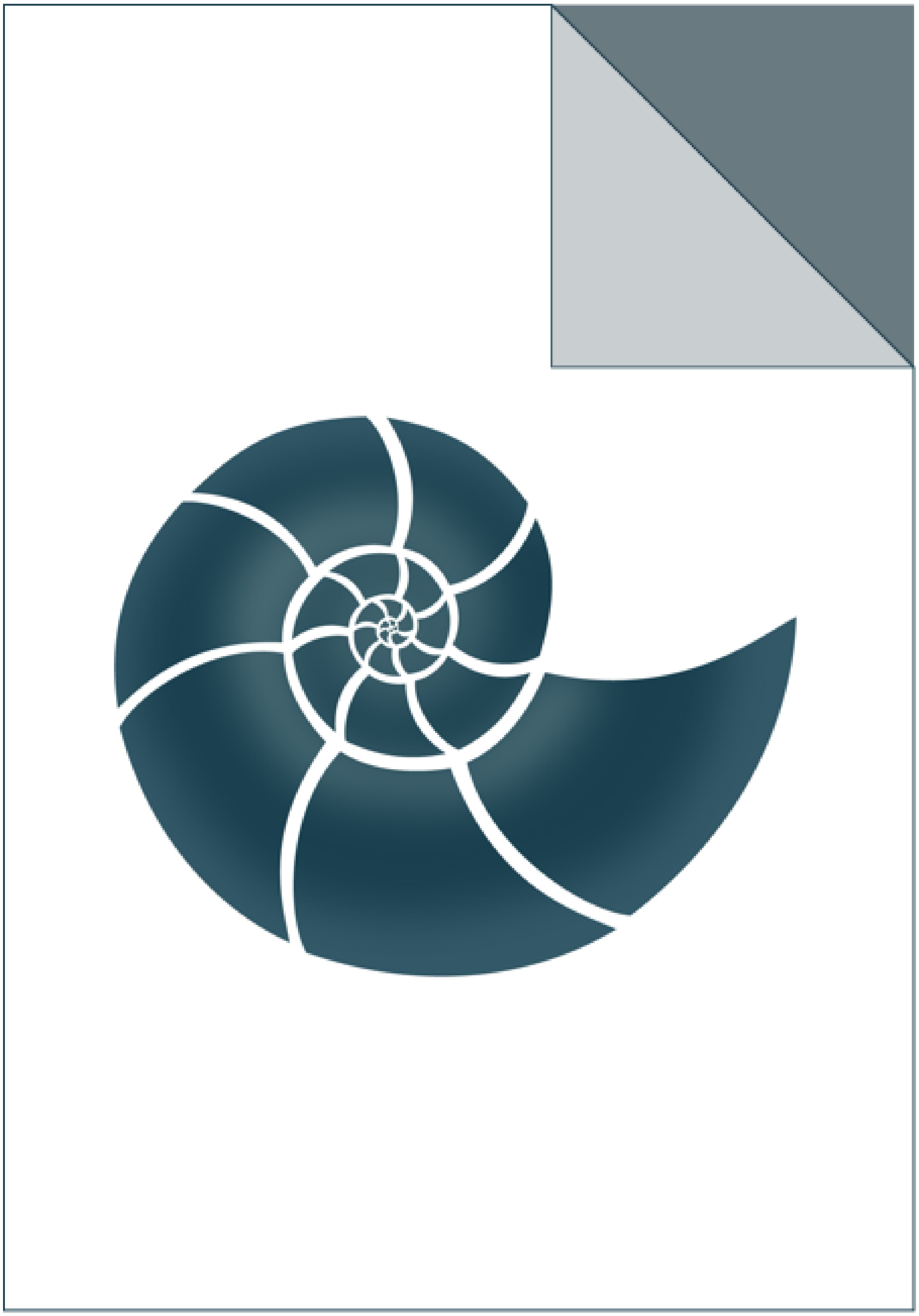
The first mandatory argument is the input MSA file, the second is the desired output file name USAGE: ./ap_SequenceProfile cyped.CYP109.aln cyped.CYP109.pro

Keywords:

- no_keywords

Categories:

- no_categories



5.1.23 ap_SequenceWeightingProtocol

> ap_SequenceWeightingProtocol reads a set of protein sequences and computes a real weight for each of those sequences.

If the FASTA file is the input, every pair of sequences will be aligned and sequence identity values will be evaluated based on these alignments. If .aln is the input (i.e. ClustalO MSA file format), it is assumed the sequences are already aligned and sequence identity values will be computed based on the MSA.

Sequence identity values will be transformed into real weights. These weights may be further used e.g. in sequence profile construction

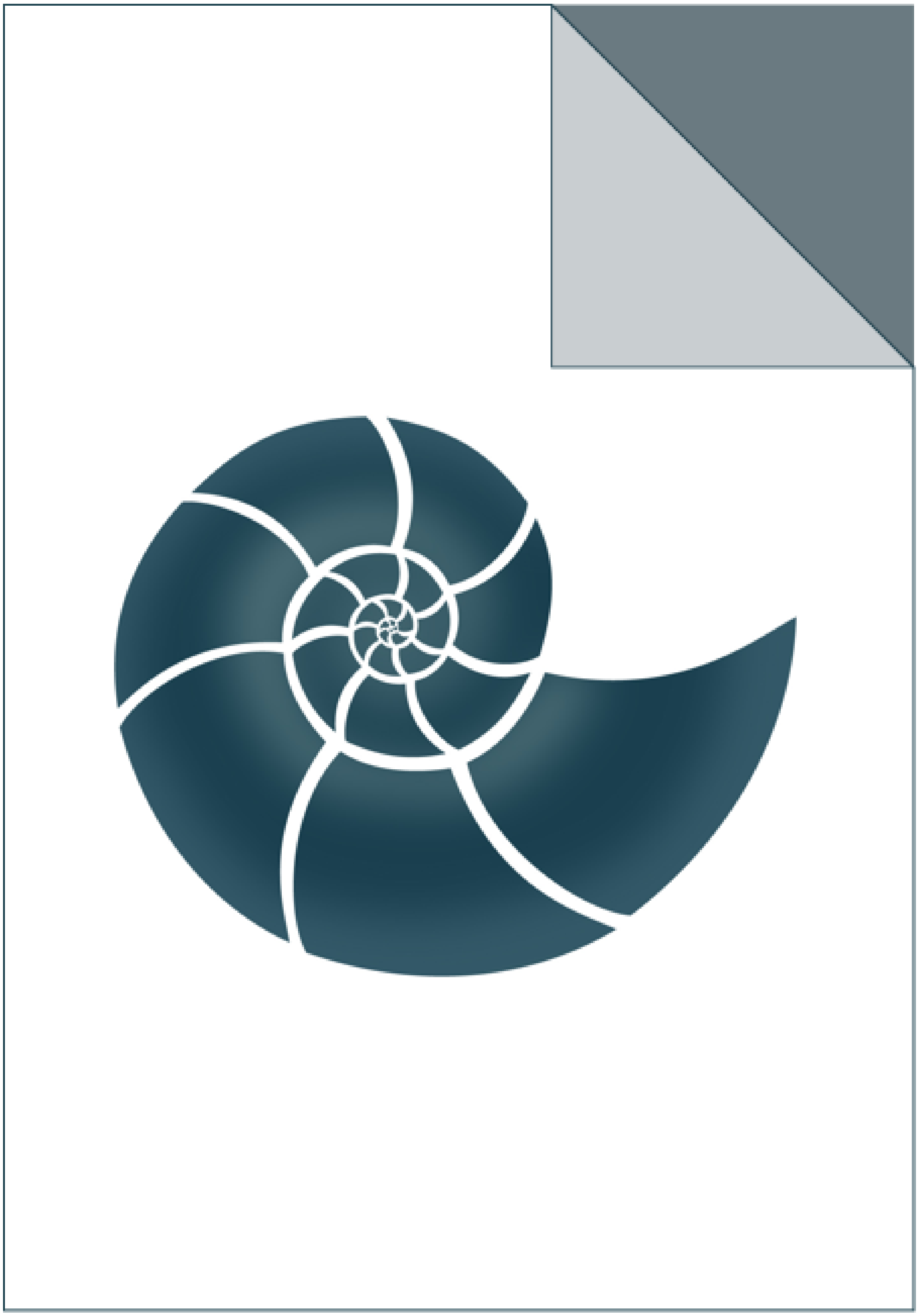
USAGE: ap_SequenceWeightingProtocol input.fasta ap_SequenceWeightingProtocol input.aln

Keywords:

- no_keywords

Categories:

- no_categories



5.1.24 ap_WeightedOnlineStatistics

> ap_WeightedOnlineStatistics reads a file with two columns: real values and their weights, and calculates their mean and stdev.

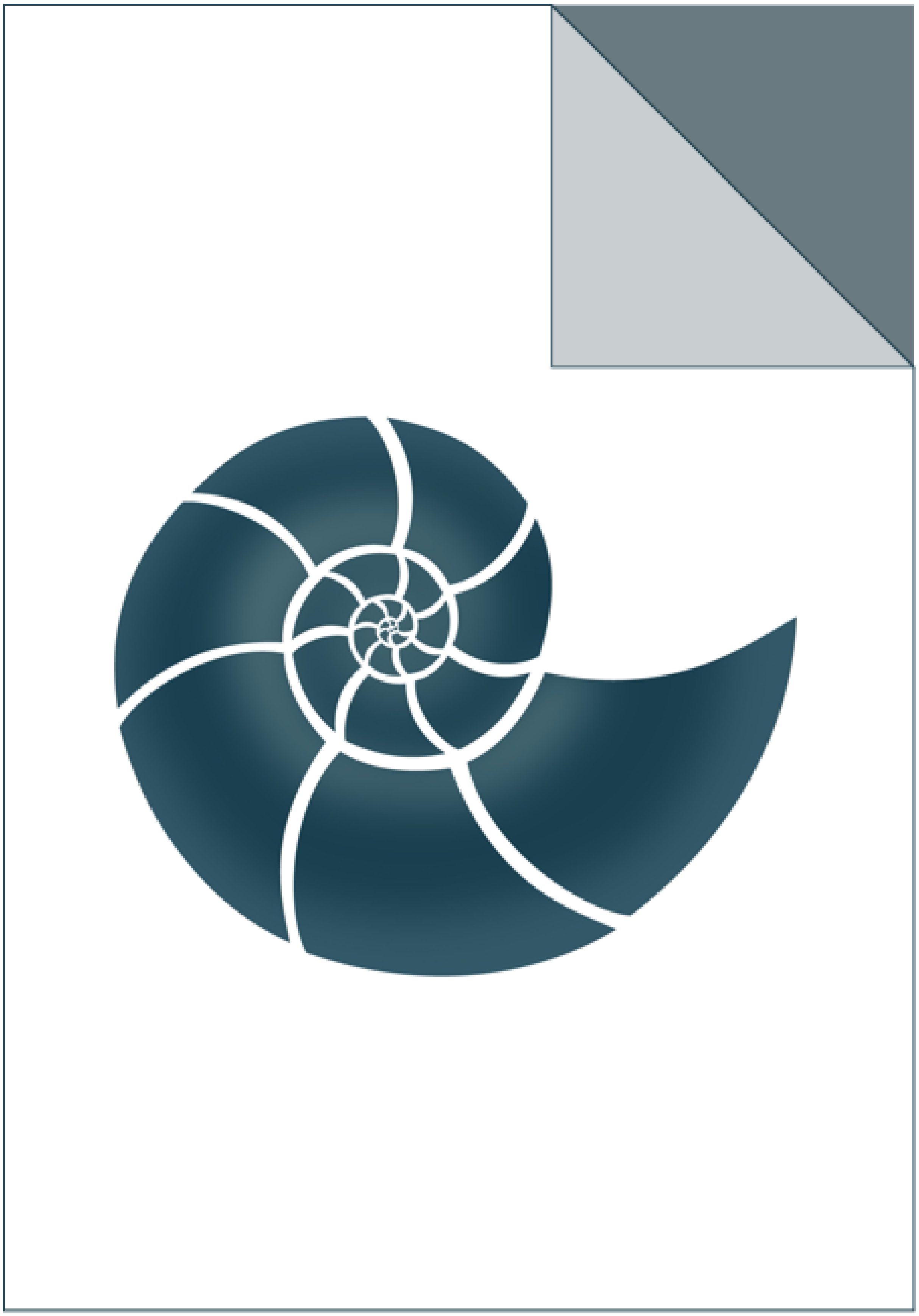
If no input file is provided, the program calculates the statistics from a random sample. USAGE: ap_WeightedOnlineStatistics infile

Keywords:

- no_keywords

Categories:

- no_categories



5.1.25 ap_align_profiles

Calculate global alignment between two sequence profiles with a gap penalty that depends on observed gap probabilities

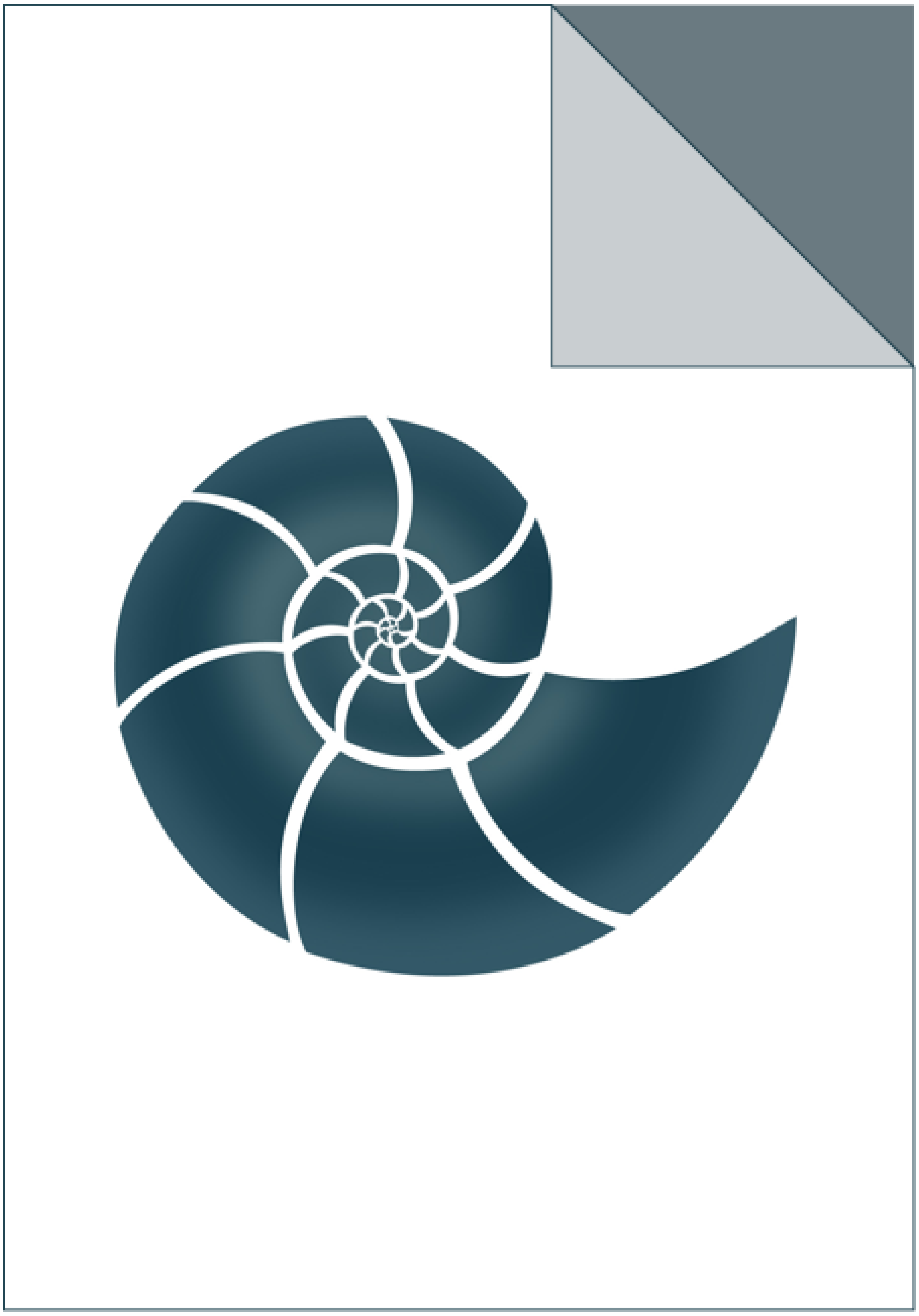
USAGE: ap_align_profiles d4proc1-A1.profile d4proc1-A2.profile [gap_open gap_extend]

Keywords:

- no_keywords

Categories:

- no_categories



5.1.26 ap_atom_correlations

ap_atom_correlations reads a multimodel PDB trajectory and calculates correlation between atomic coordinates

USAGE: ap_atom_correlations 2kwi.pdb

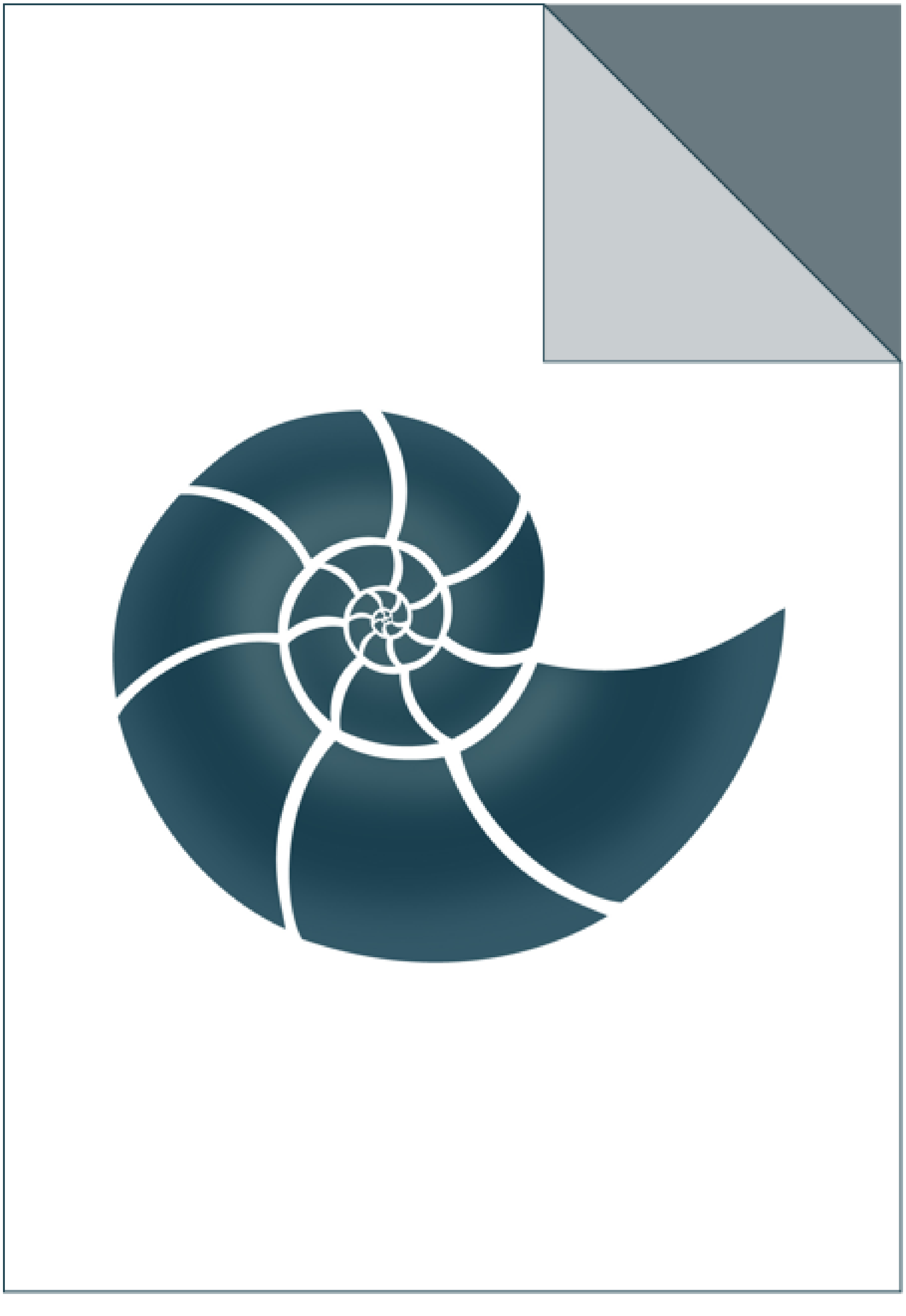
where 2kwi.pdb id an input file

Keywords:

- no_keywords

Categories:

- no_categories



5.1.27 ap_blastxml_to_fasta

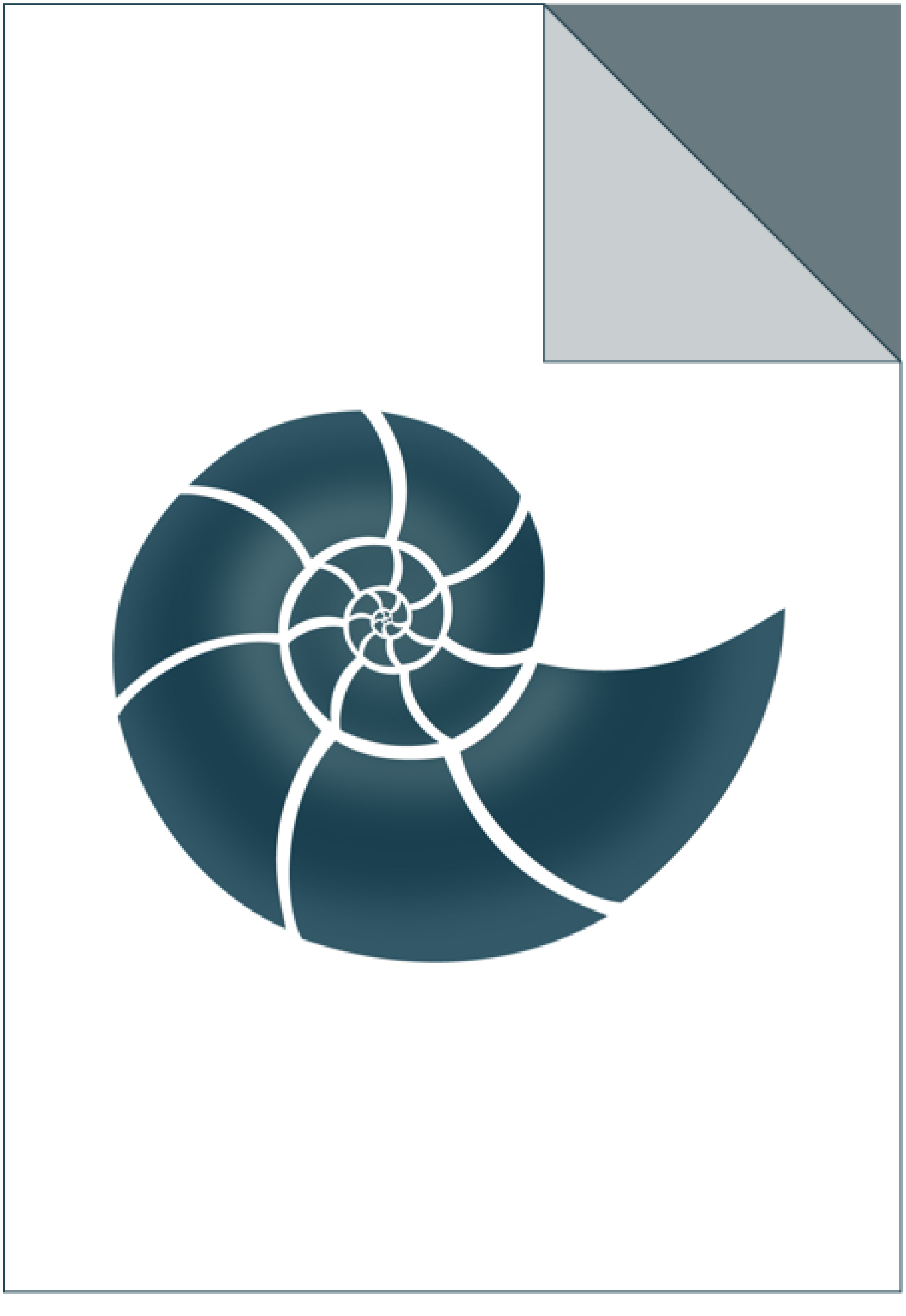
Reads XML produced by psiblast and creates FASTA file containing all hits

Keywords:

- no_keywords

Categories:

- no_categories



5.1.28 ap_build_crystal

ap_create_crystal reads a given PDB file and prints all atoms in a unit cell.

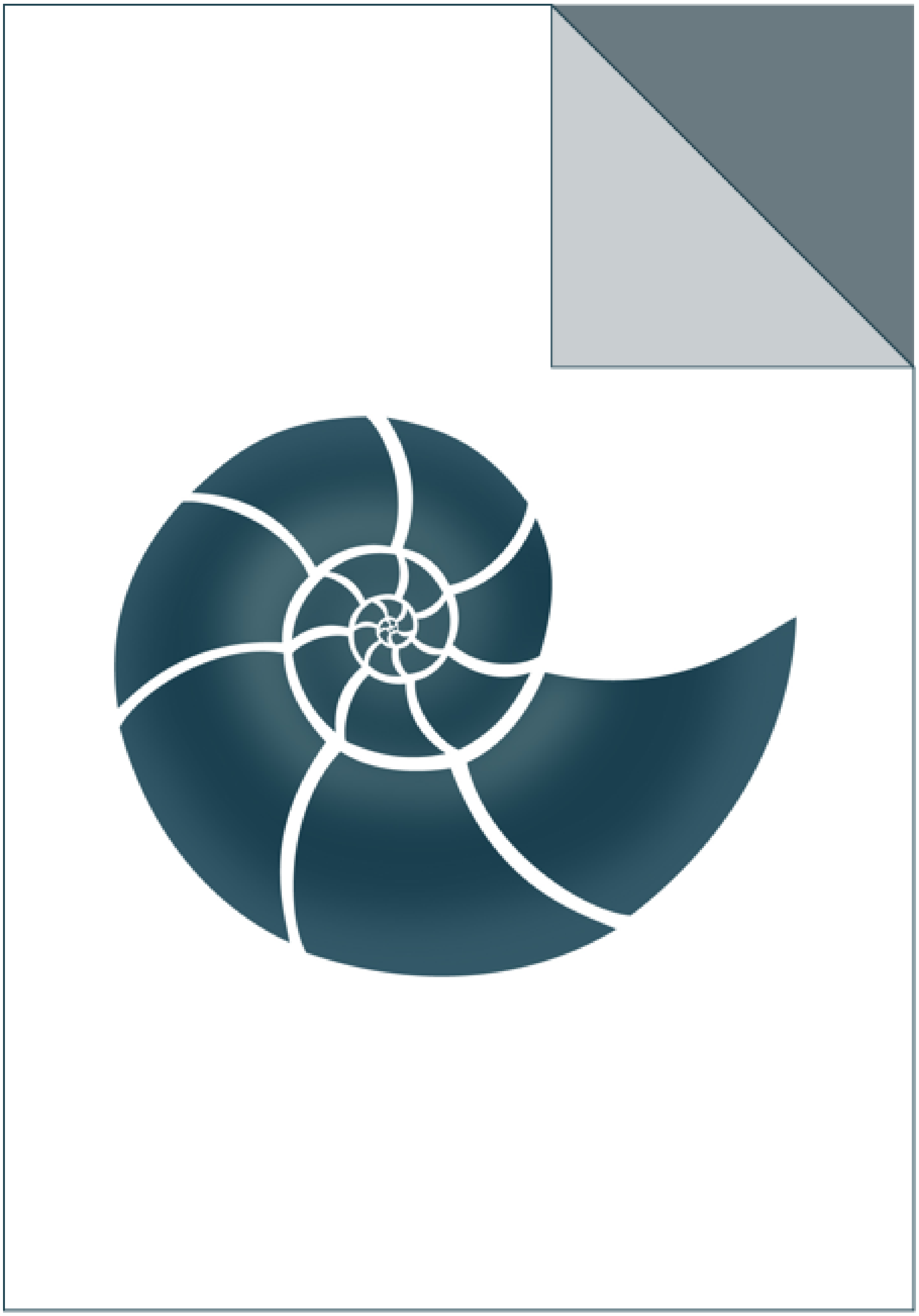
USAGE: ap_create_crystal 5edw.pdb

Keywords:

- no_keywords

Categories:

- no_categories



5.1.29 ap_contact_map_overlap

ap_contact_map_overlap calculates overlap between contact maps calculated for two (or more) structures

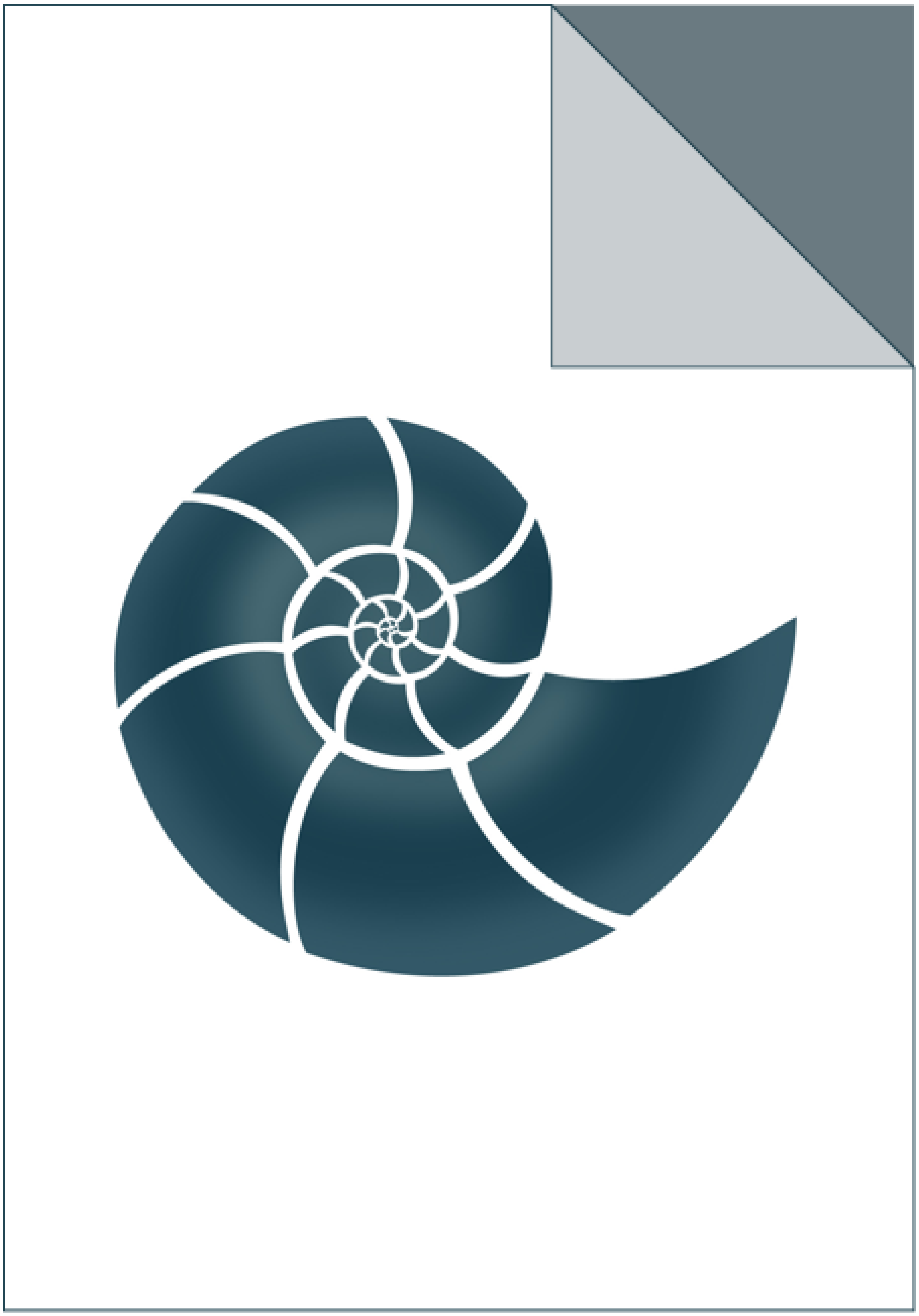
USAGE: ap_contact_map_overlap CA native.pdb models.pdb cutoff 8.0 In this case the program evaluates contact map overlap (measured by Jaccard coefficient) between the native structure and every model found in models.pdb. 8.0 is the contact distance in Angstroms and CA defines the contact map type; allowed options are: CA CB and SC for Calpha, C-beta and all atom side chain, respectively

Keywords:

- no_keywords

Categories:

- no_categories



5.1.30 ap_docking_crmsd

> ap_docking_crmsd calculates crmsd between ligand positions after flexible docking to a receptor and a reference.

The program reads in a native pose and at least one PDB file with a computed pose (i.e. a model), each of them must contain a ligand molecule bound to a protein receptor. The ligand can be a small molecule, peptide or even a protein. The program finds the ligand either by residue ID (a three-letter code, such as CAM) or a chain ID - a single letter code.

USAGE: ap_docking_crmsd 2m56-ref.pdb CAM 00199.pdb 00963.pdb 04473.pdb ap_docking_crmsd 2m56-ref.pdb X 00199.pdb 00963.pdb 04473.pdb ap_docking_crmsd - X 00199.pdb 00963.pdb 04473.pdb

where 2m56-ref.pdb is the native and CAM is the three-letter PDB code of the ligand for which crmsd will be evaluated and 00199.pdb and the two other files are conformation after docking. In the second example, X is the ID of the chain containing a ligand molecule.

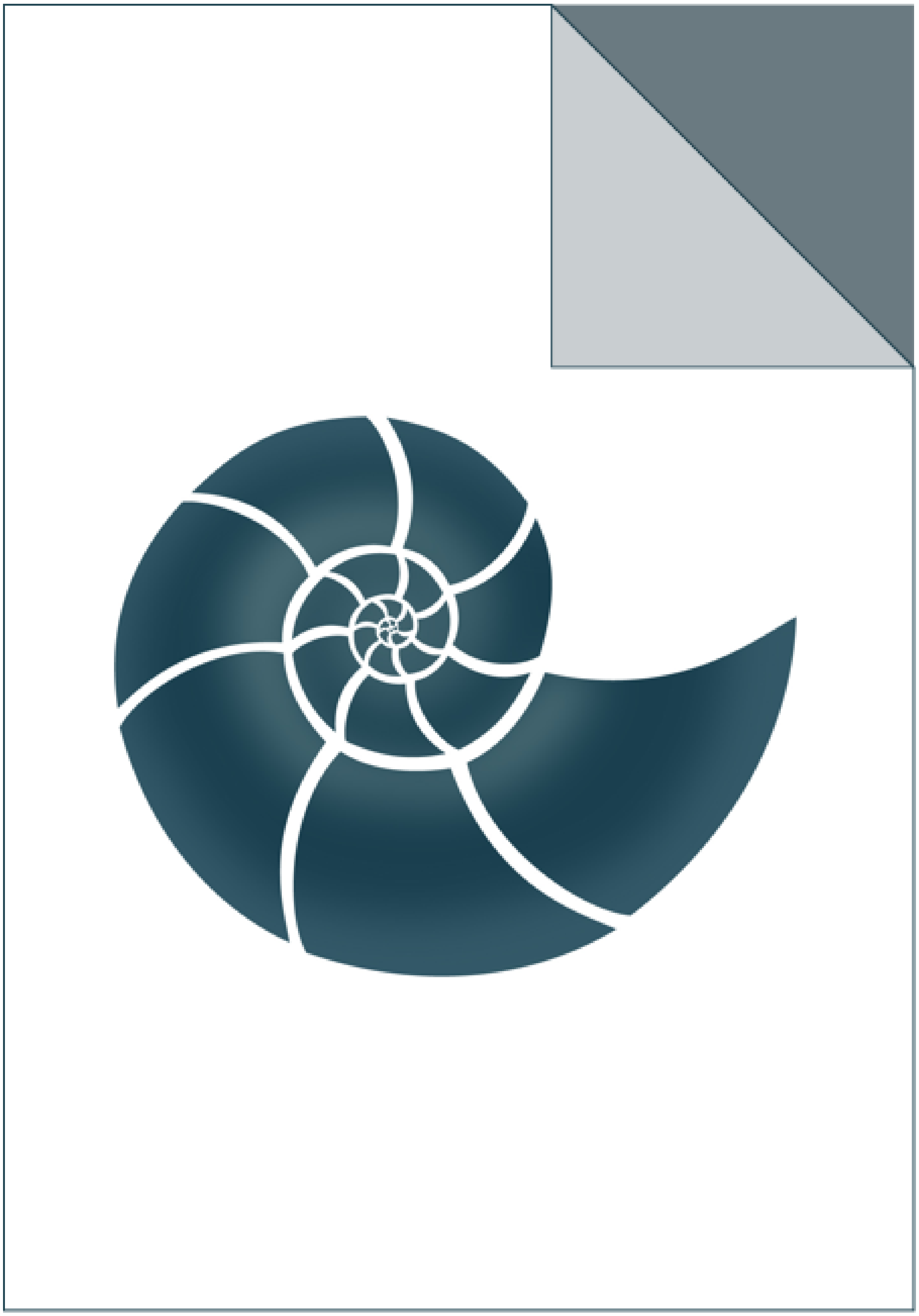
The program evaluates crmsd based on ligand cooridinates upon an optimal superimposition of a receptor between the refence and any model of any input file. If the reference structure is not given and dash '-' character is used instead (as in the last example), the program evaluates pairwise all-vs-all crmsd calculations

Keywords:

- no_keywords

Categories:

- no_categories



5.1.31 ap_download_pdb

Simple app downloads a pdb file from RCSB website

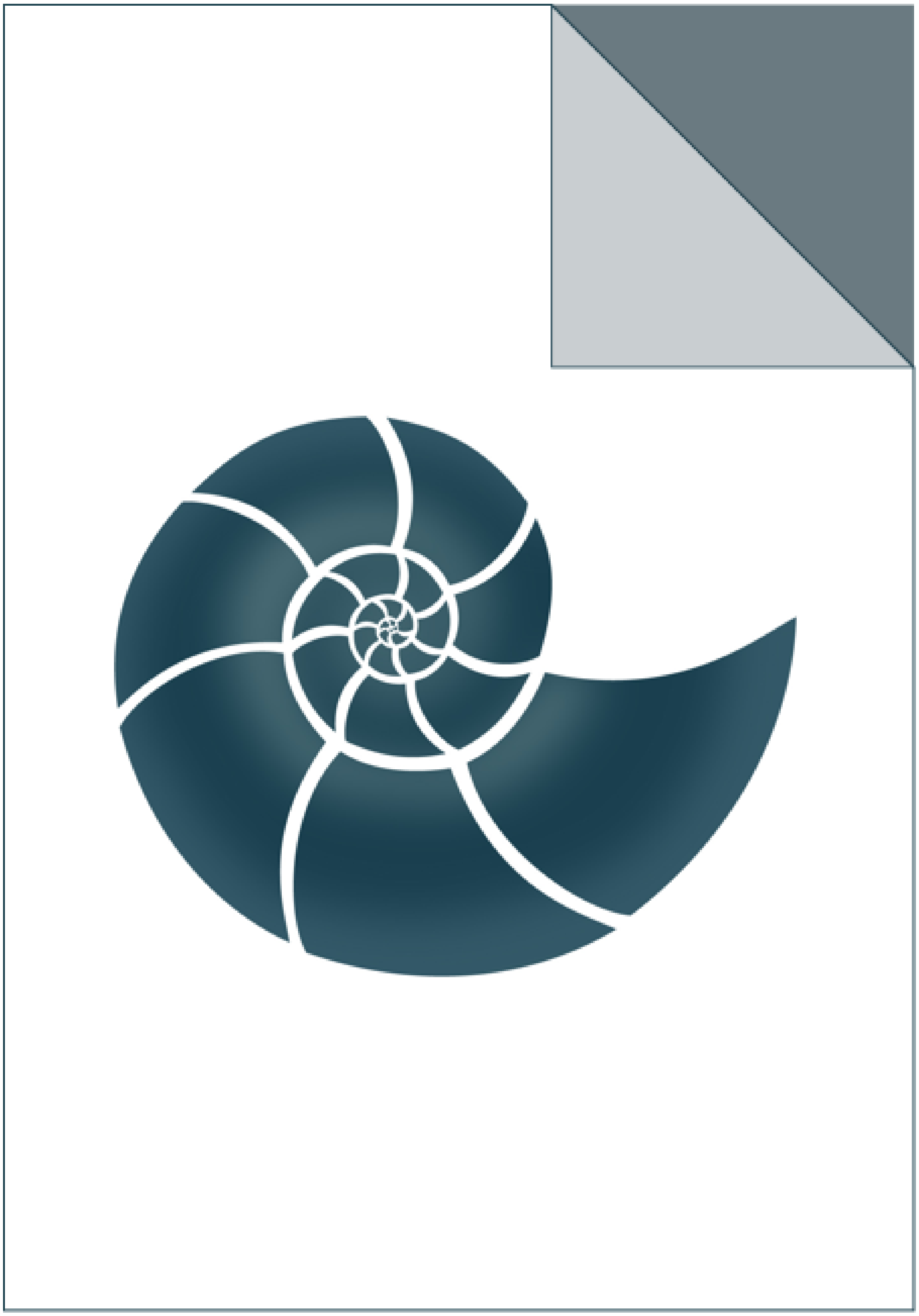
USAGE: ap_download_pdb PDB_code

Keywords:

- no_keywords

Categories:

- no_categories



5.1.32 ap_dssp

Detects secondary structure using BioShell's implementation of the DSSP algorithm.

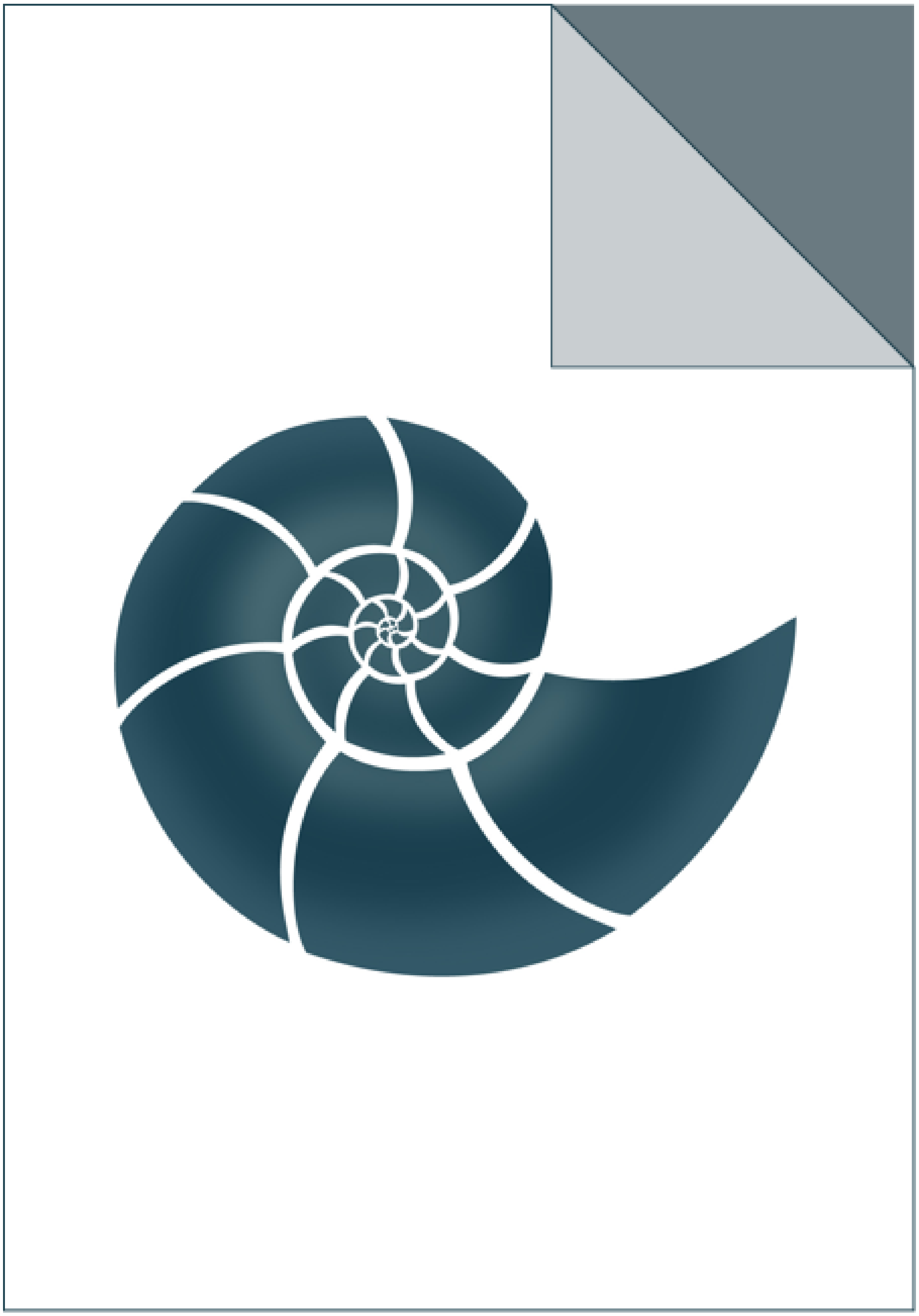
USAGE: ap_dssp 5edw.pdb

Keywords:

- no_keywords

Categories:

- no_categories



5.1.33 ap_dssp_to_ss2

ap_dssp_to_ss2 reads a DSSP file and writes secondary structure in SS2 format

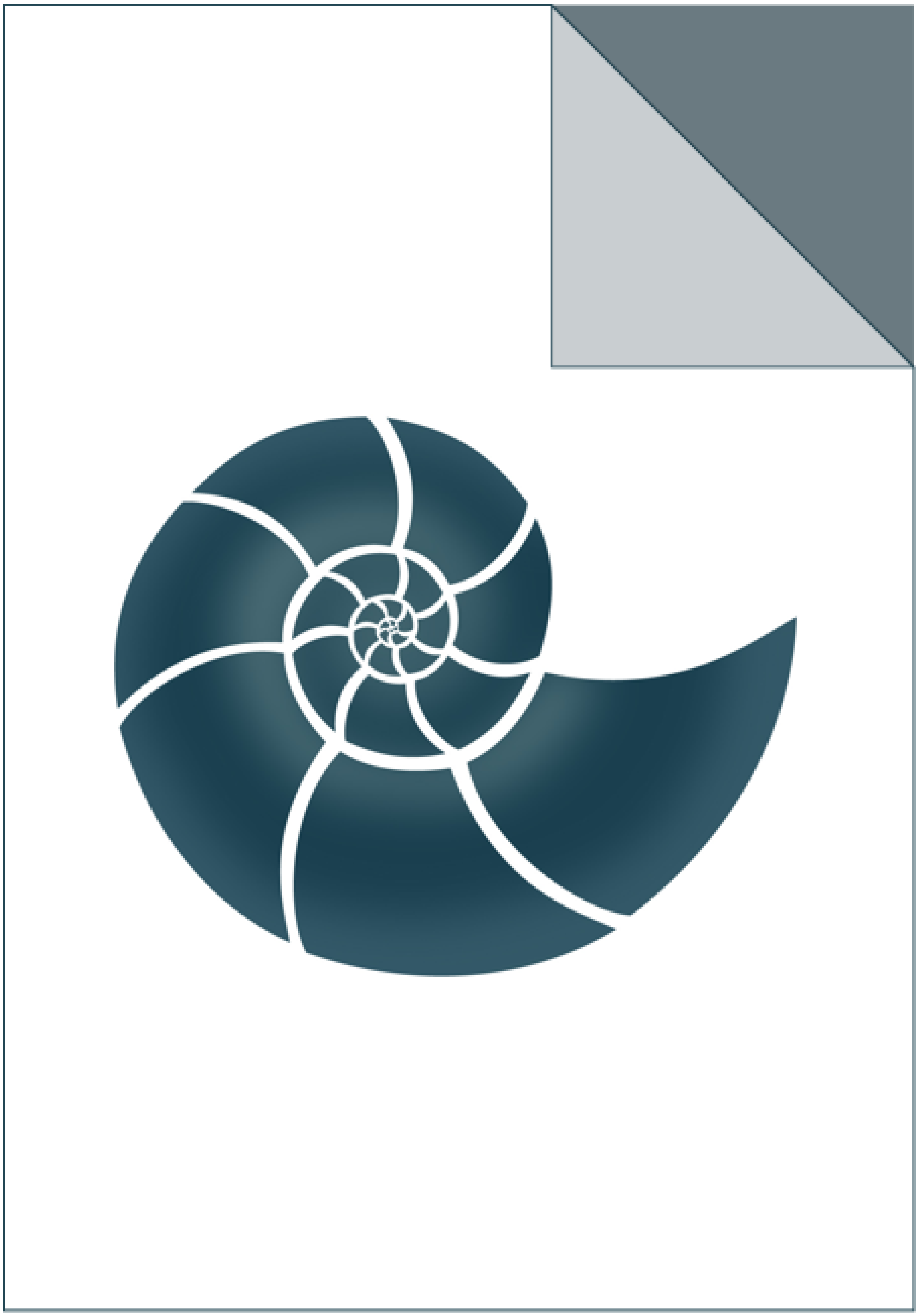
USAGE: ap_dssp_to_ss2 5edw.dssp

Keywords:

- no_keywords

Categories:

- no_categories



5.1.34 ap_filter_fasta

ap_find_in_fasta reads a file in FASTA format and prints only these sequences which satisfy the following filters:

- sequence must a protein
- sequence must not be shorter than 15 aa
- sequence must contain at most 10 UNK residues

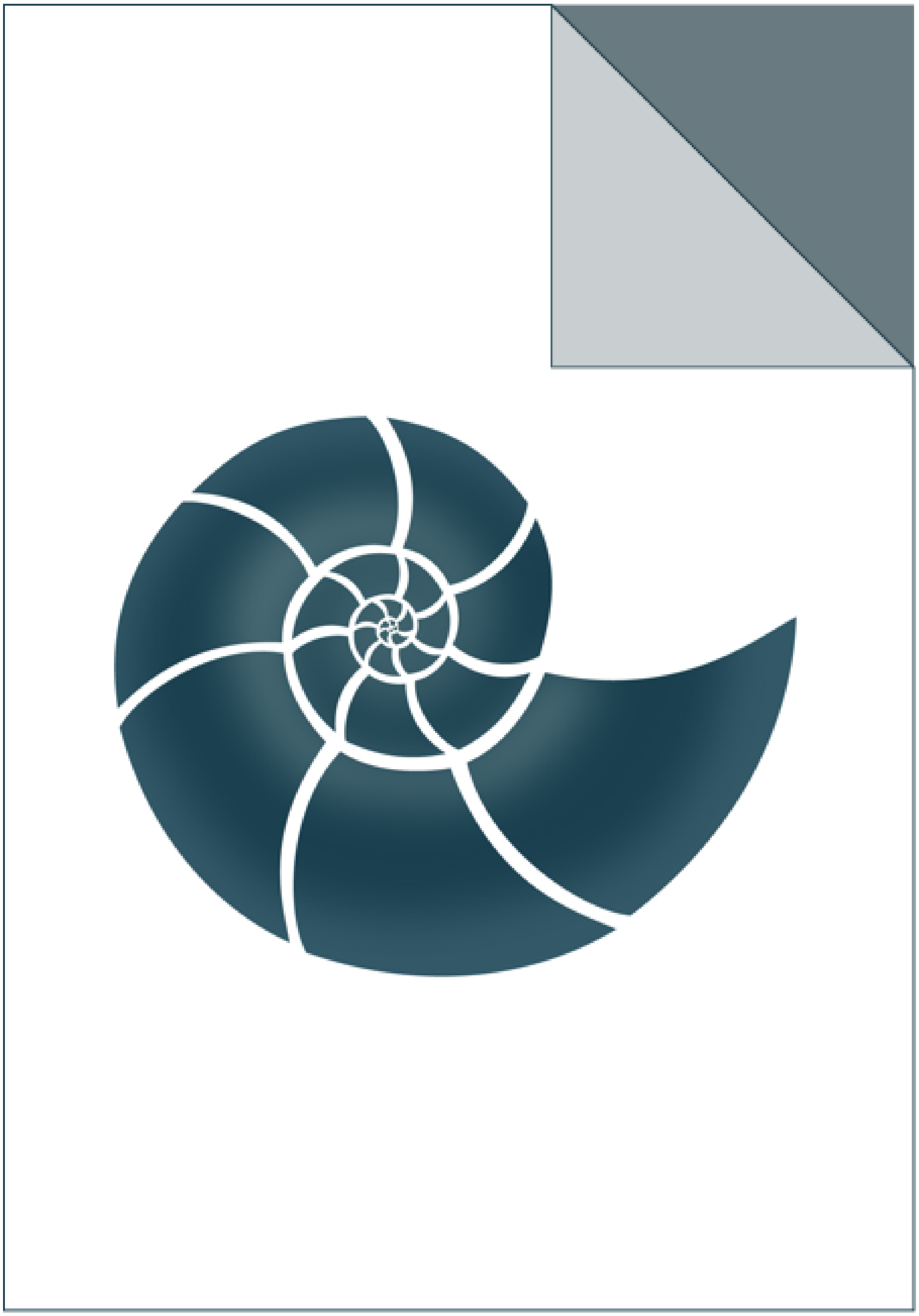
The output sequences are sorted. USAGE: ap_filter_fasta input.fasta [input2.fasta . . .]

Keywords:

- no_keywords

Categories:

- no_categories



5.1.35 ap_find_in_fasta

ap_find_in_fasta reads a sequence database in FASTA format and looks for sequences by given IDs

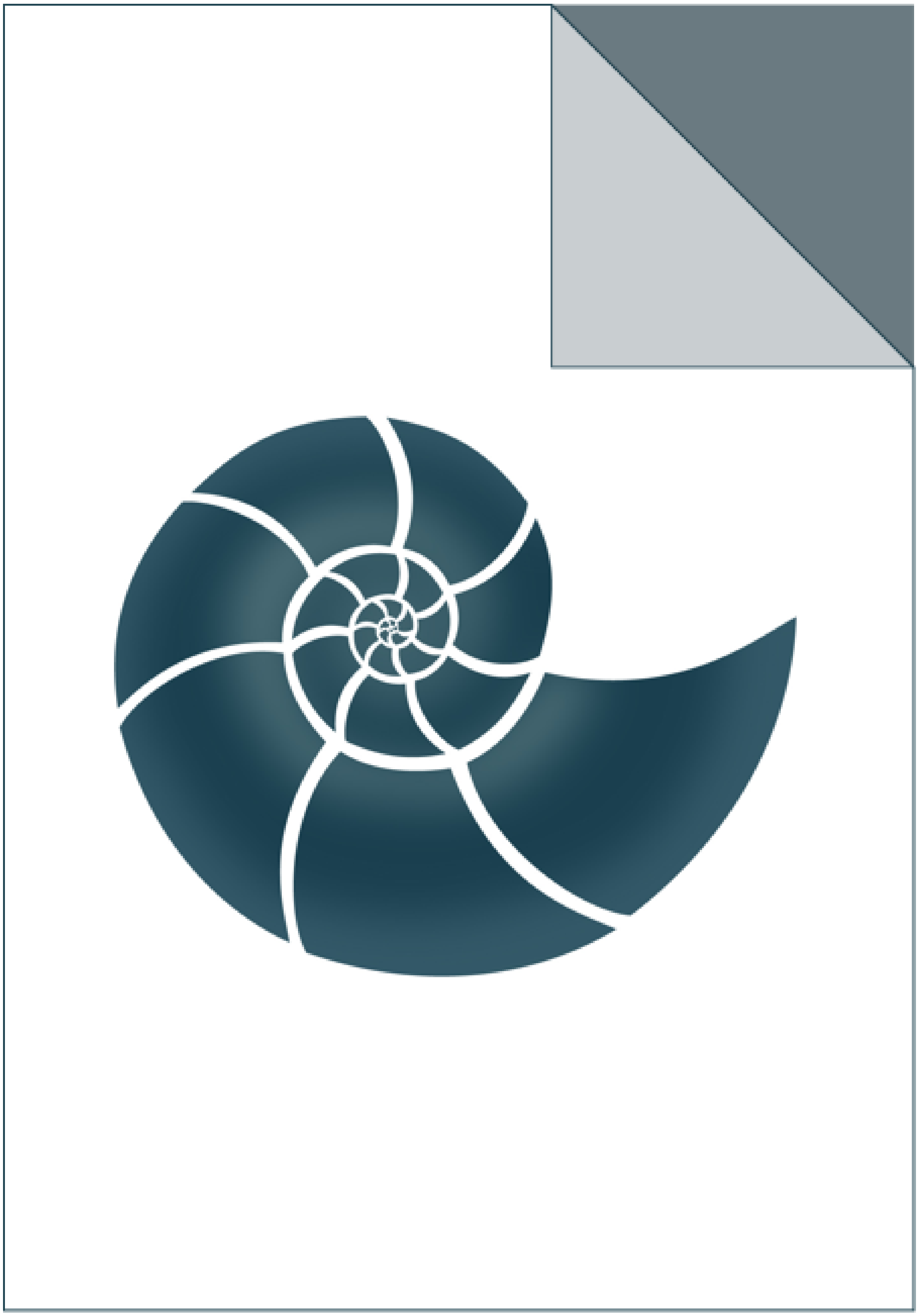
USAGE: ap_find_in_fasta uniref90.fasta seq_id_list.txt

Keywords:

- no_keywords

Categories:

- no_categories



5.1.36 ap_ligand_trajectory

ap_ligand_trajectory finds contacts between a ligand molecule and a protein.

It reads a multi-model PDB file and detects contacts in every model (e.g. frame). The output provides the interacting residues (name and residueId) along with the number of observations for this contact.

USAGE: ap_ligand_trajectory 2kwi.pdb GNP 3.5

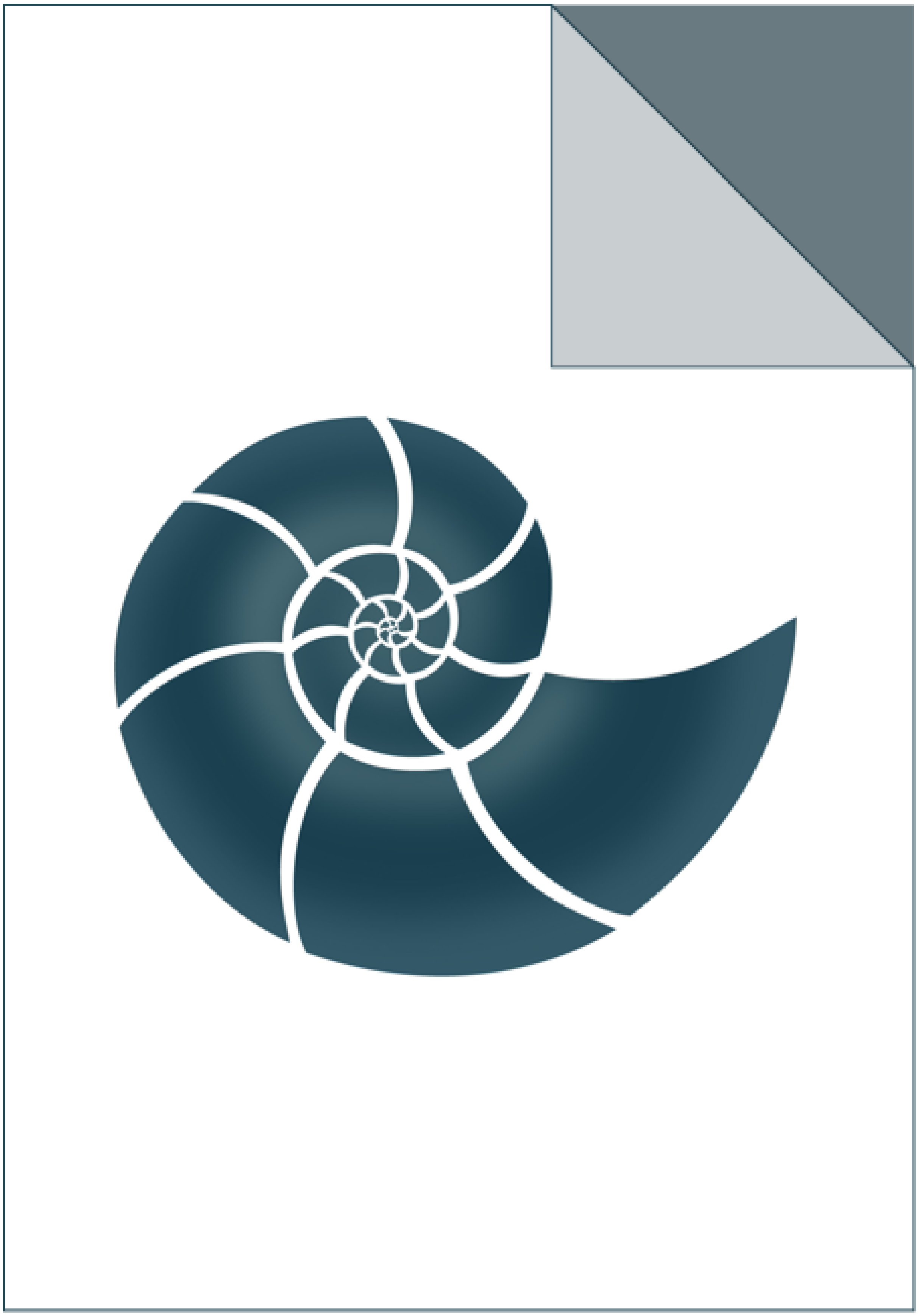
where 2kwi.pdb id an input file, GNP the ligand code and 3.5 - contact distance in Angstroms

Keywords:

- no_keywords

Categories:

- no_categories



5.1.37 ap_local_backbone_geometry

Program reads a protein structure (PDB format) and calculates local backbone properties: distances, angles, etc.

The list of requested properties should follow the PDB innput file name. If no properties are listed at command line, the program calculates all known properties.

USAGE: ./ap_local_backbone_geometry input.pdb property1 property2 … EXAMPLE: ./ap_local_backbone_geometry 2gb1.pdb PHI PSI OMEGA ./ap_local_backbone_geometry 2gb1.pdb

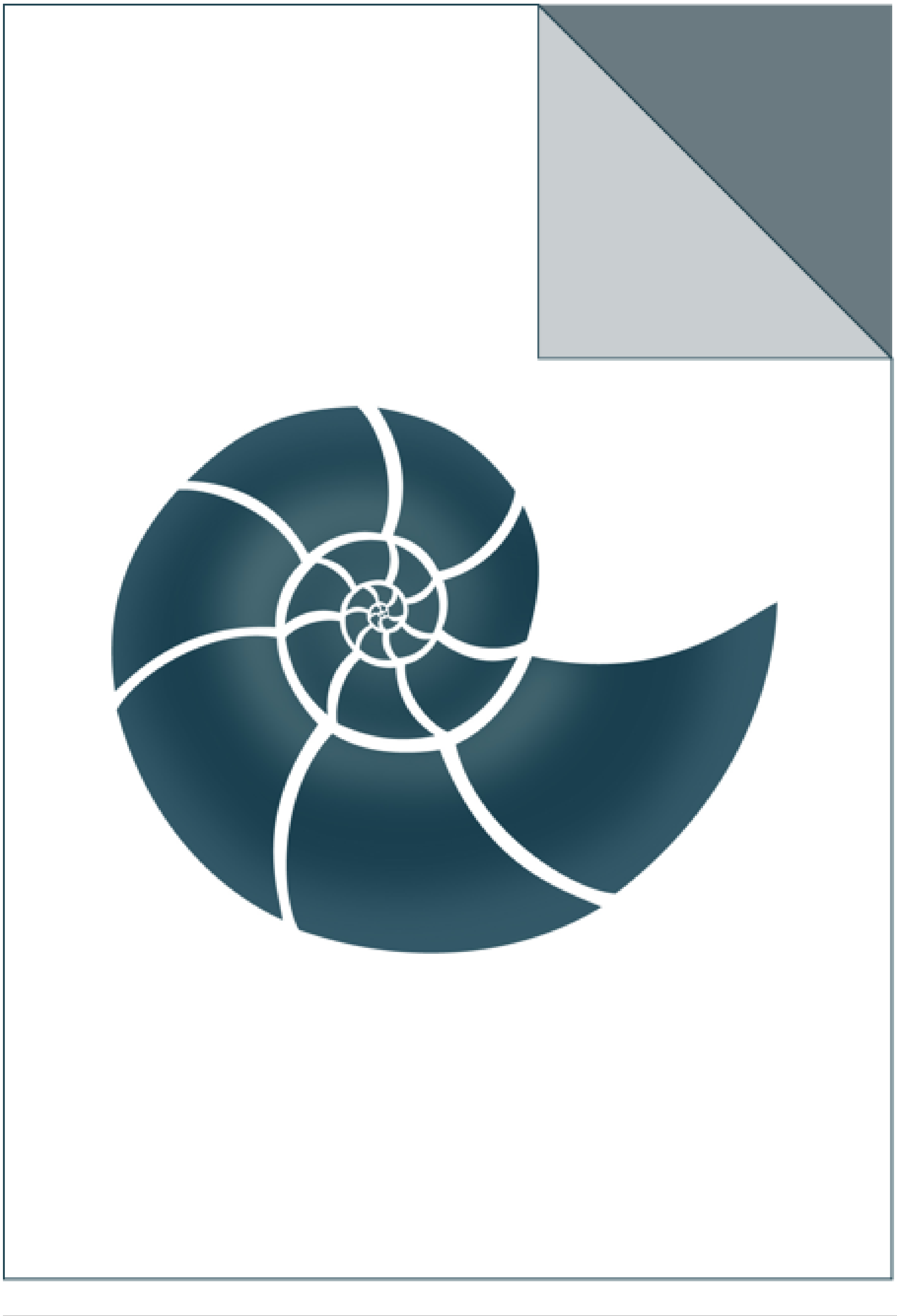
Known properties

Keywords:

- no_keywords

Categories:

- no_categories



5.1.38 ap_molecule_diffusion

> ap_molecule_diffusion calculates average displacement of a small molecule as a function of time over a trajectory

If a multi-model PDB file was given, the program prints contact count observed in all models

USAGE: ap_molecule_diffusion trajectory.pdb HOH box_side

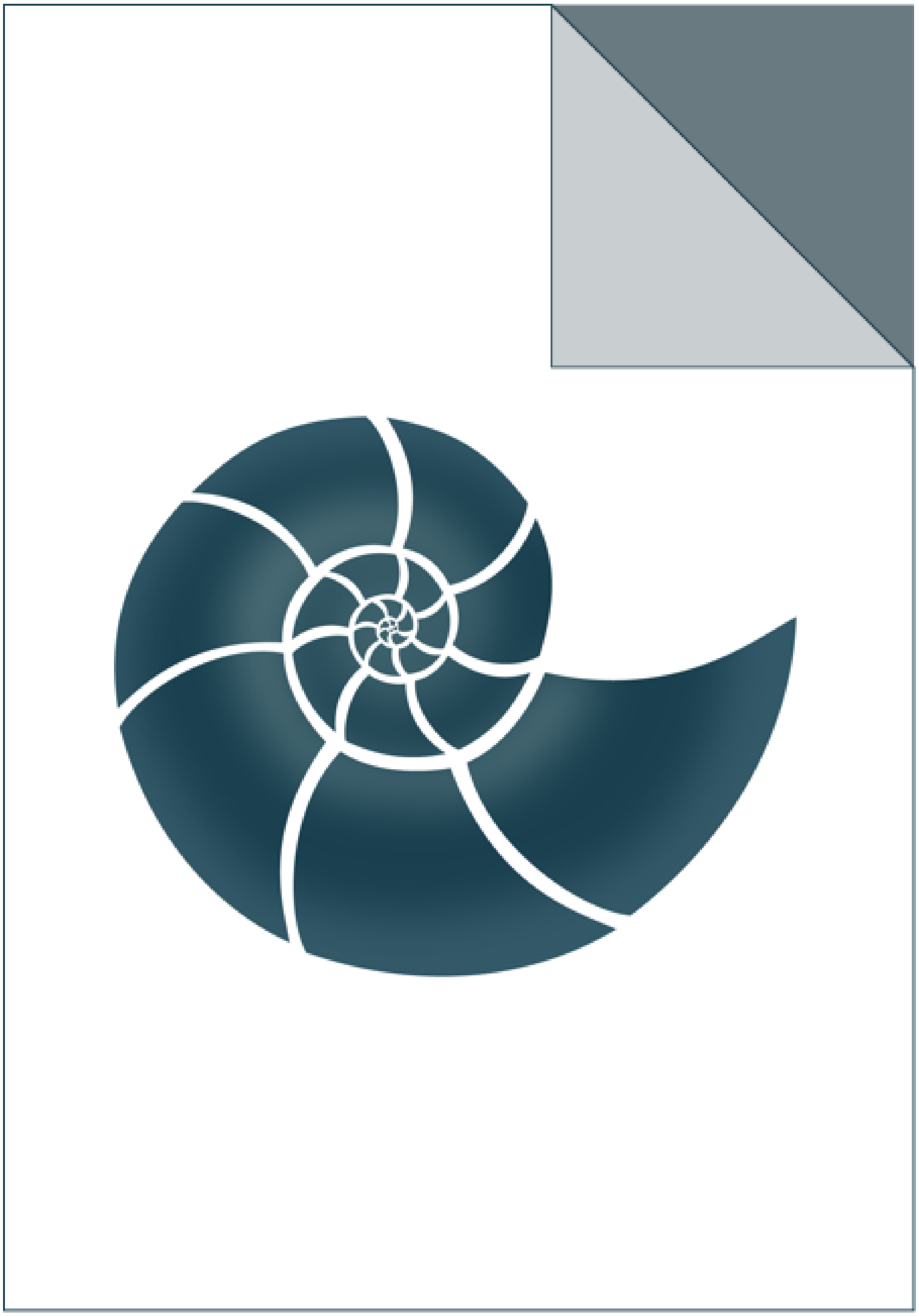
where trajectory.pdb is the input file multimodel-PDB file HOH is the PDB-id of molecules for which the displacement will be evaluated

Keywords:

- no_keywords

Categories:

- no_categories



5.1.39 ap_pdb_to_fasta_ss

Reads a PDB file and writes protein sequence(s) in FASTA format.

The program also writes secondary structure in FASTA format, if this data is available from PDB headers. The sequence comprise only these amino acid residues which have C-alpha atom User can select a chain by providing its code as the second argument of the program. The program also writes PDB file that corresponds to the sequence.

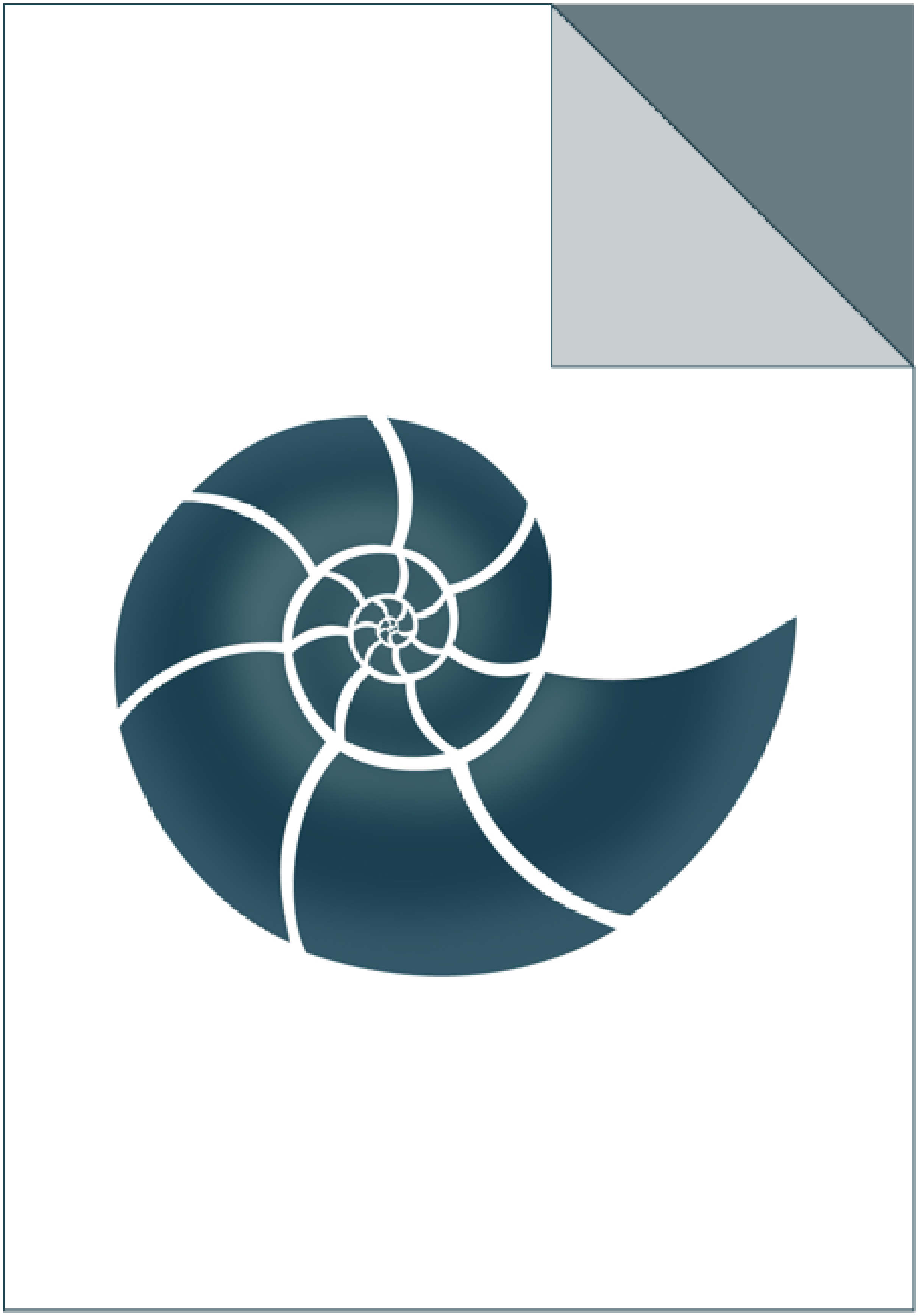
USAGE: ap_pdb_to_fasta_ss 5edw.pdb A

Keywords:

- no_keywords

Categories:

- no_categories



5.1.40 ap_pdb_to_pir

Reads a PDB file and writes protein sequence(s) in PIR format

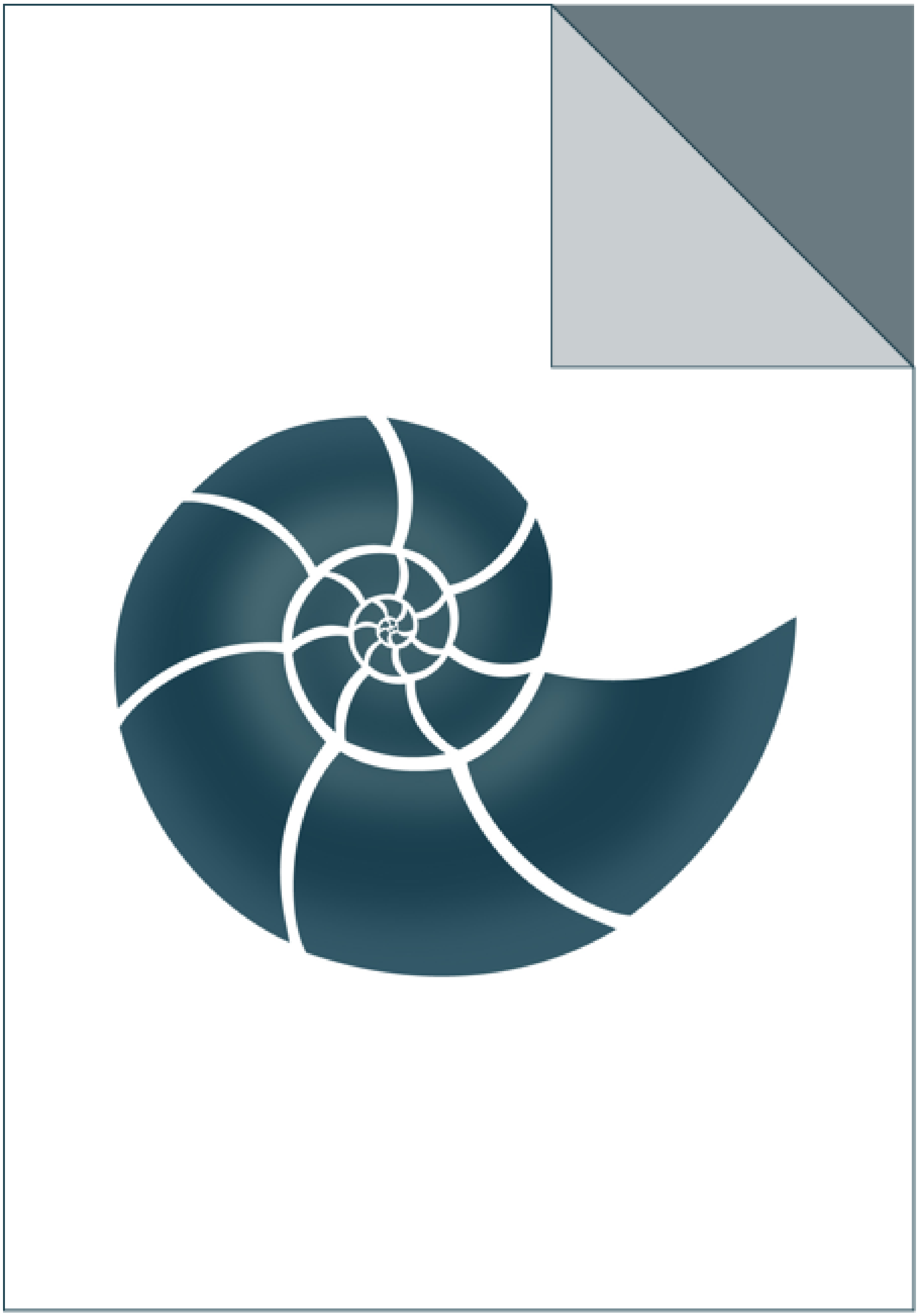
USAGE: ap_pdb_to_pir 5edw.pdb

Keywords:

- no_keywords

Categories:

- no_categories



5.1.41 ap_pir_to_fasta

Reads a file with sequences in PIR format and converts them to FASTA.

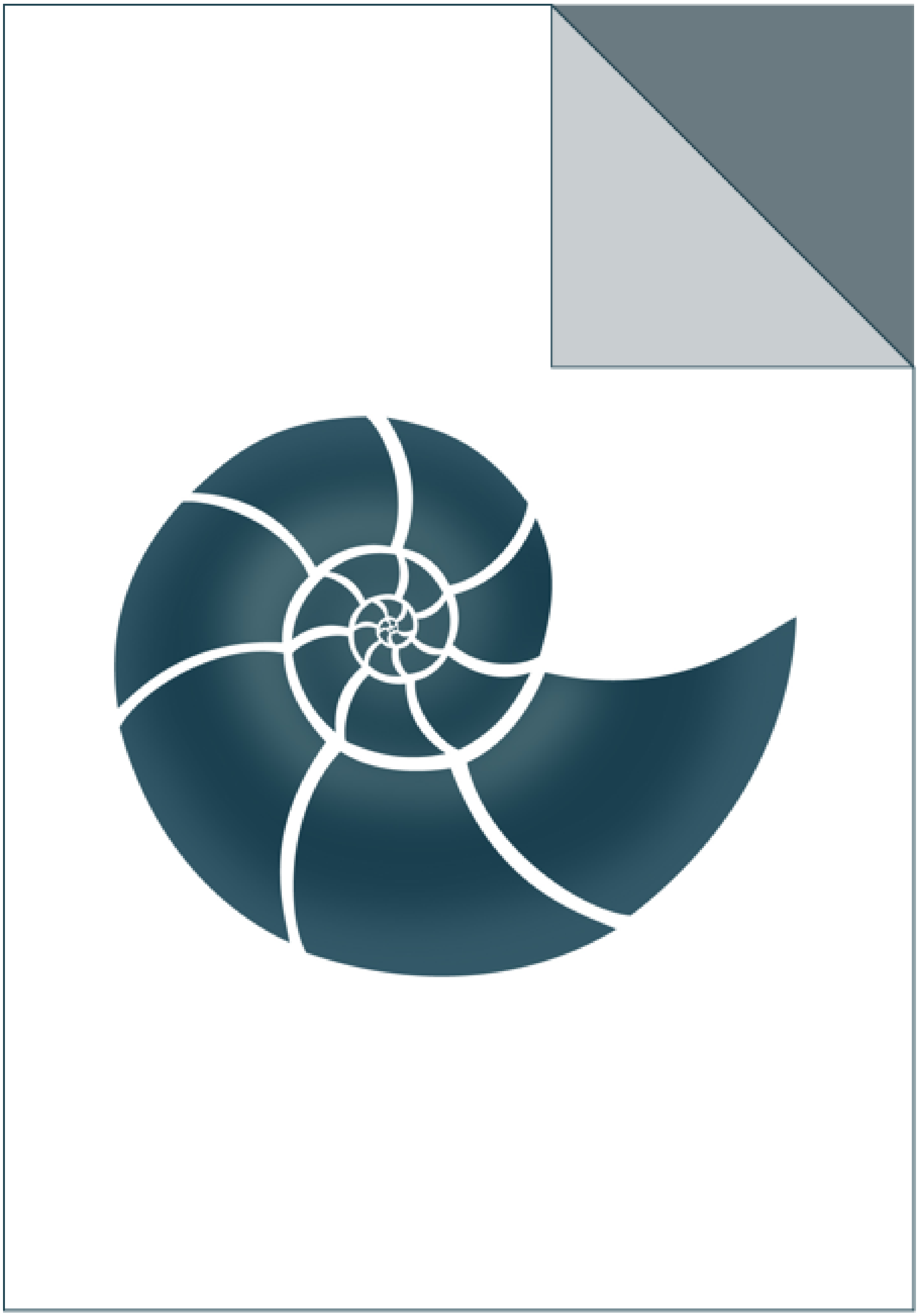
USAGE: ap_pir_to_fasta example.pir

Keywords:

- no_keywords

Categories:

- no_categories



5.1.42 ap_reorder_profile_columns

ap_reorder_profile_columns reads a sequence profile (ASN.1 file format) and shuffles profile's columns as requested.

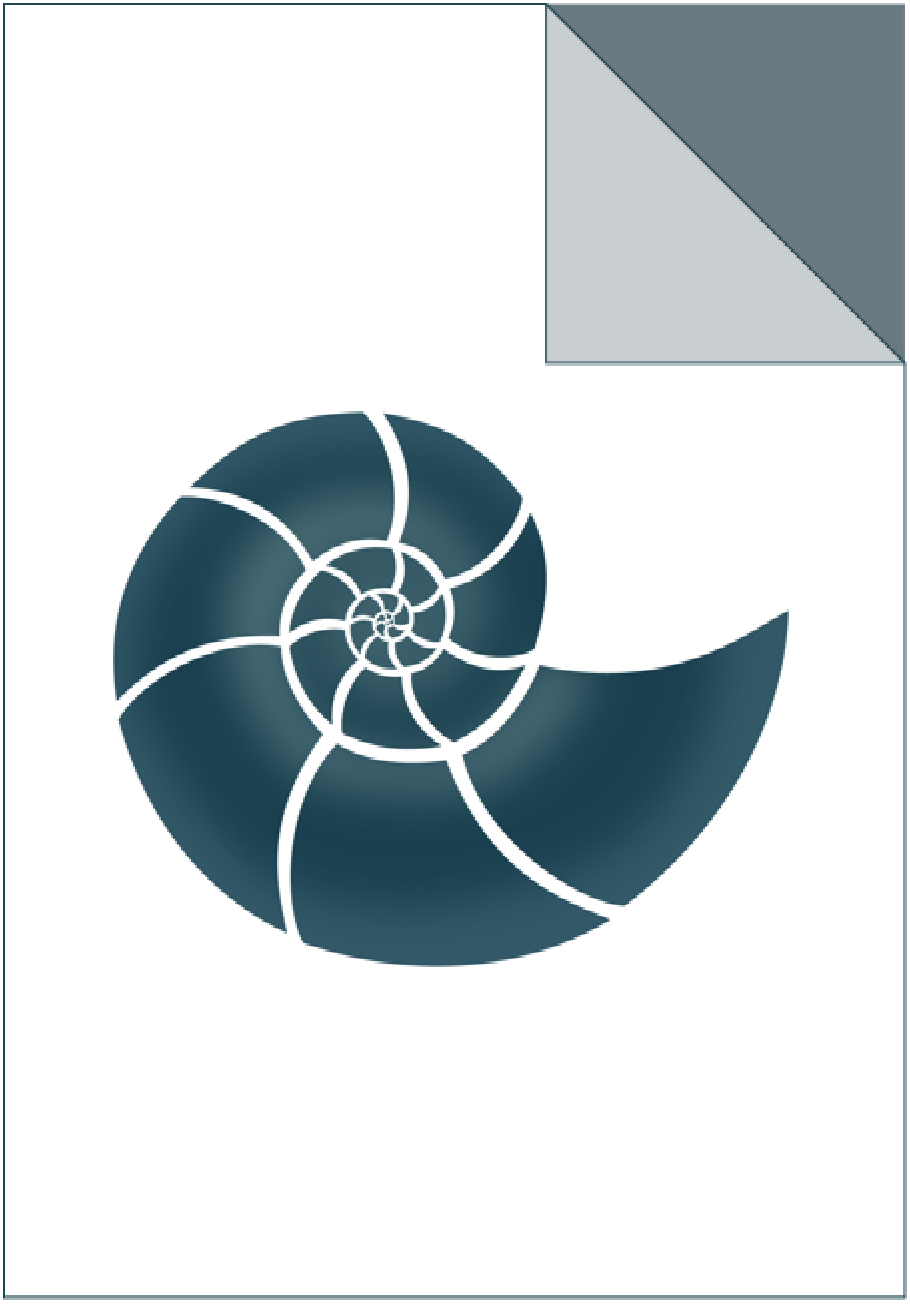
Resulting profile is writen in text format USAGE: ./ap_reorder_profile_columns input.asn1

Keywords:

- no_keywords

Categories:

- no_categories



5.1.43 ap_rescore_alignment

Reads in a sequence alignment in a FASTA format and recalculates its score. By default it uses

BLOSUM62 substitution matrix with -10 and -1 as gap opening and gap extension penalty, respectively. These parameters may be changed from command line (optional parameters)

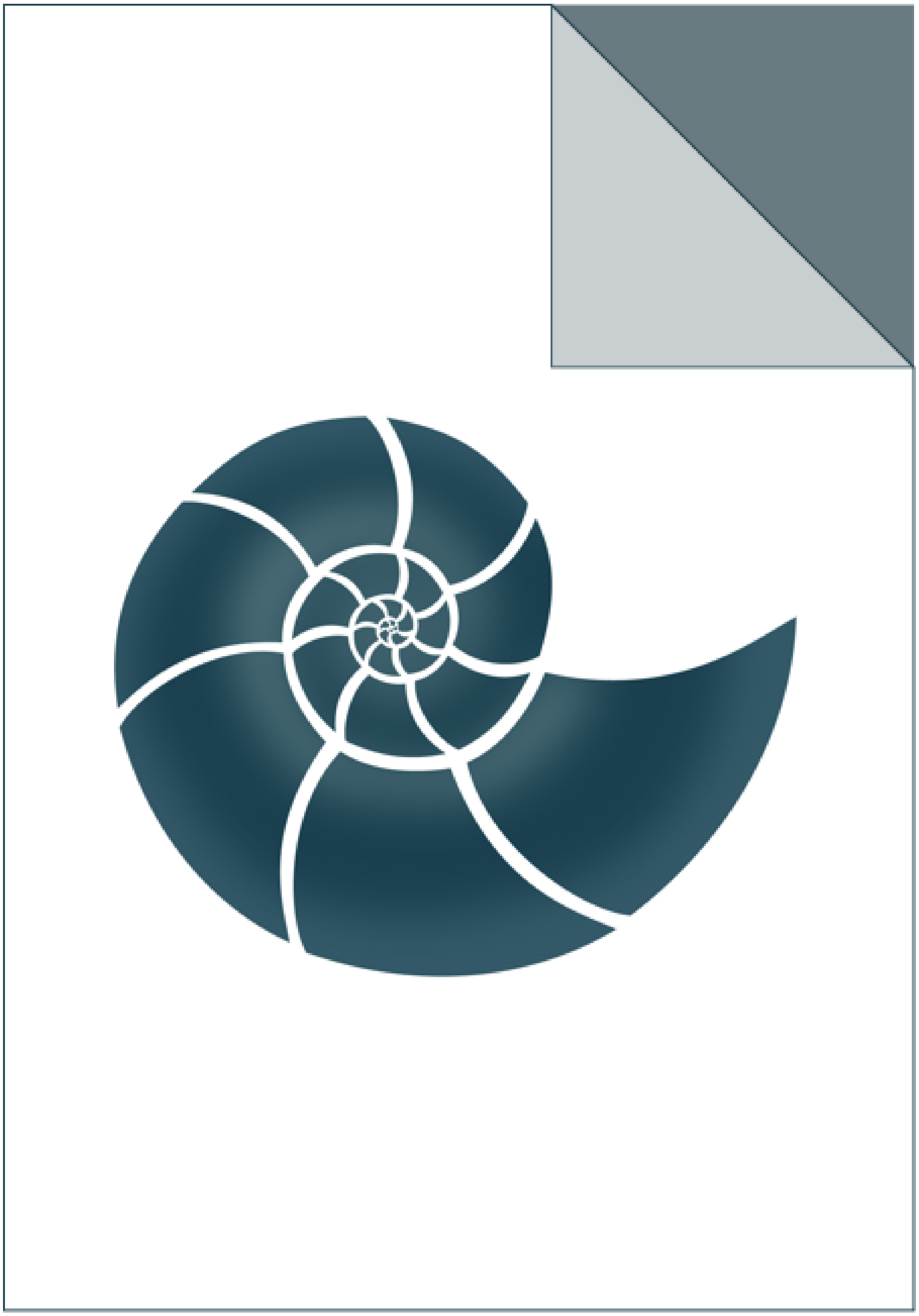
USAGE: ap_rescore_alignment ali.fasta [BLOSUM62 -10 -1]

Keywords:

- no_keywords

Categories:

- no_categories



5.1.44 ap_scorefile_columns

Reads a score file or a silent file (produced by Rosetta) and extracts requested columns of scores

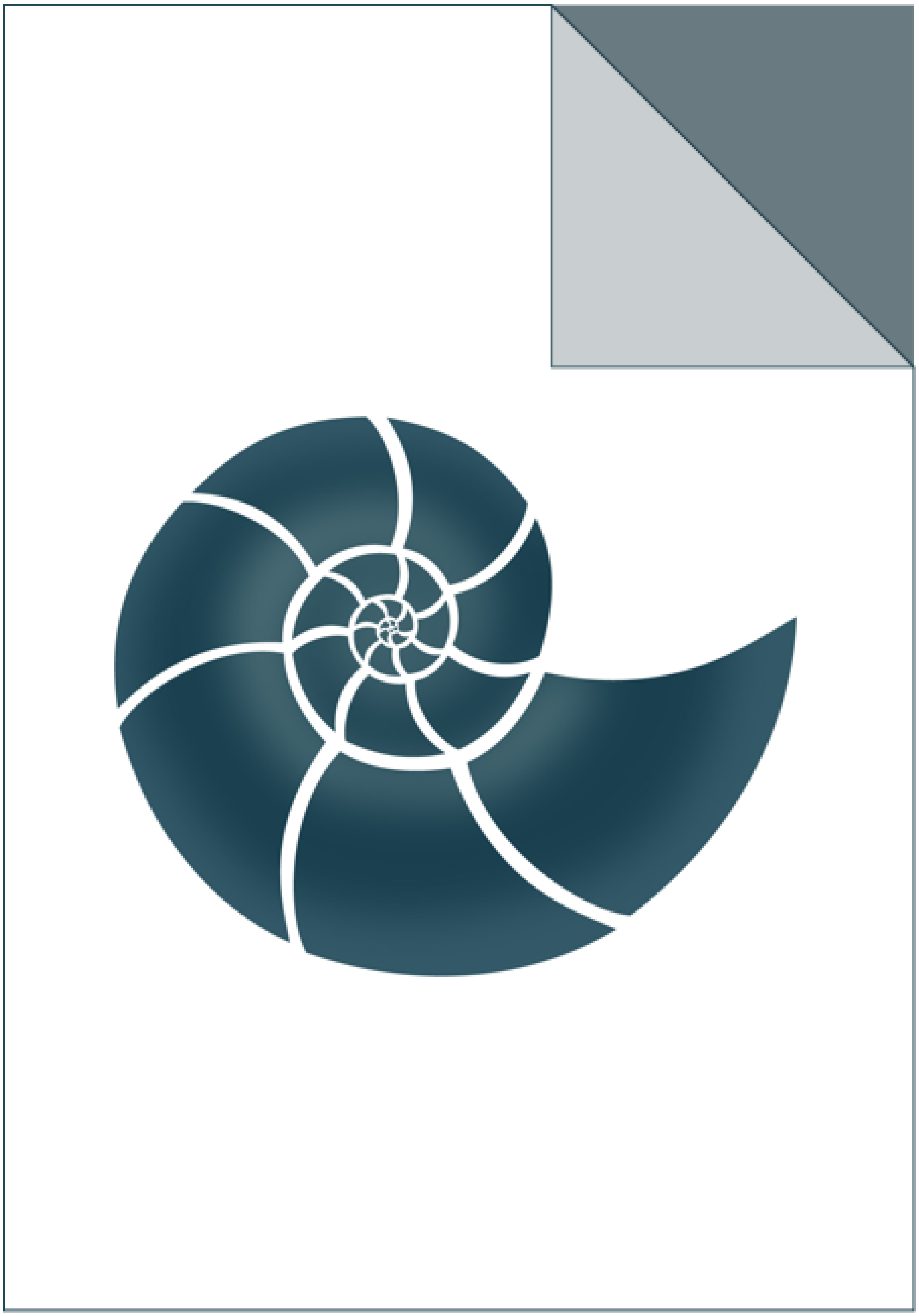
USAGE: ap_scorefile_columns default.out ap_scorefile_columns score.fsc

Keywords:

- no_keywords

Categories:

- no_categories



5.1.45 ap_stacking_interactions

Finds stacking interactions in a given PDB file.

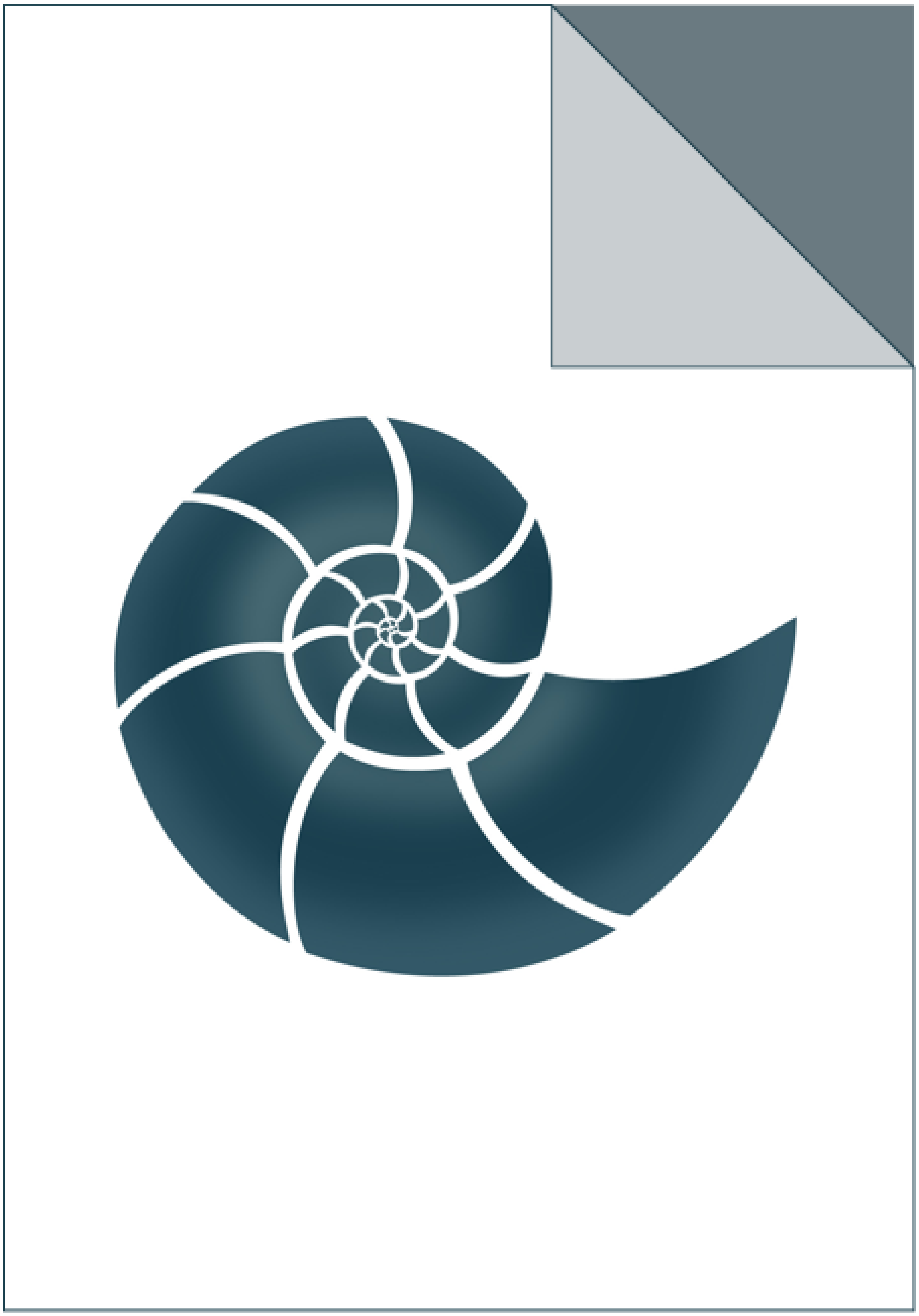
The program prints relative orientation (three Euler angles and the distance) between any two aromatic rings found in amino acid side chains that are closer than 5.0. The rings are assumed to be flat rigid moieties. USAGE: ap_stacking_interactions 5edw.pdb

Keywords:

- no_keywords

Categories:

- no_categories



5.1.46 ap_stiff_docking_crmsd

> ap_stiff_docking_crmsd calculates crmsd of a ligand that is bound to a receptor, assuming the receptor conformation has not changed much

The program reads in a native pose and at least one PDB file with a computed pose (i.e. a model), each of them must contain a ligand molecule bound to a protein receptor. The ligand can be a small molecule, peptide or even a protein. The program finds the ligand either by residue ID (a three-letter code, such as CAM) or a chain ID - a single letter code.

USAGE: ap_stiff_docking_crmsd 2m56-ref.pdb CAM 00199.pdb 00963.pdb 04473.pdb ap_stiff_docking_crmsd 2m56-ref.pdb X 00199.pdb 00963.pdb 04473.pdb ap_stiff_docking_crmsd - X 00199.pdb 00963.pdb 04473.pdb

where 2m56-ref.pdb is the native and CAM is the three-letter PDB code of the ligand for which crmsd will be evaluated and 00199.pdb and the two other files are conformation after docking. In the second example, X is the ID of the chain containing a ligand molecule.

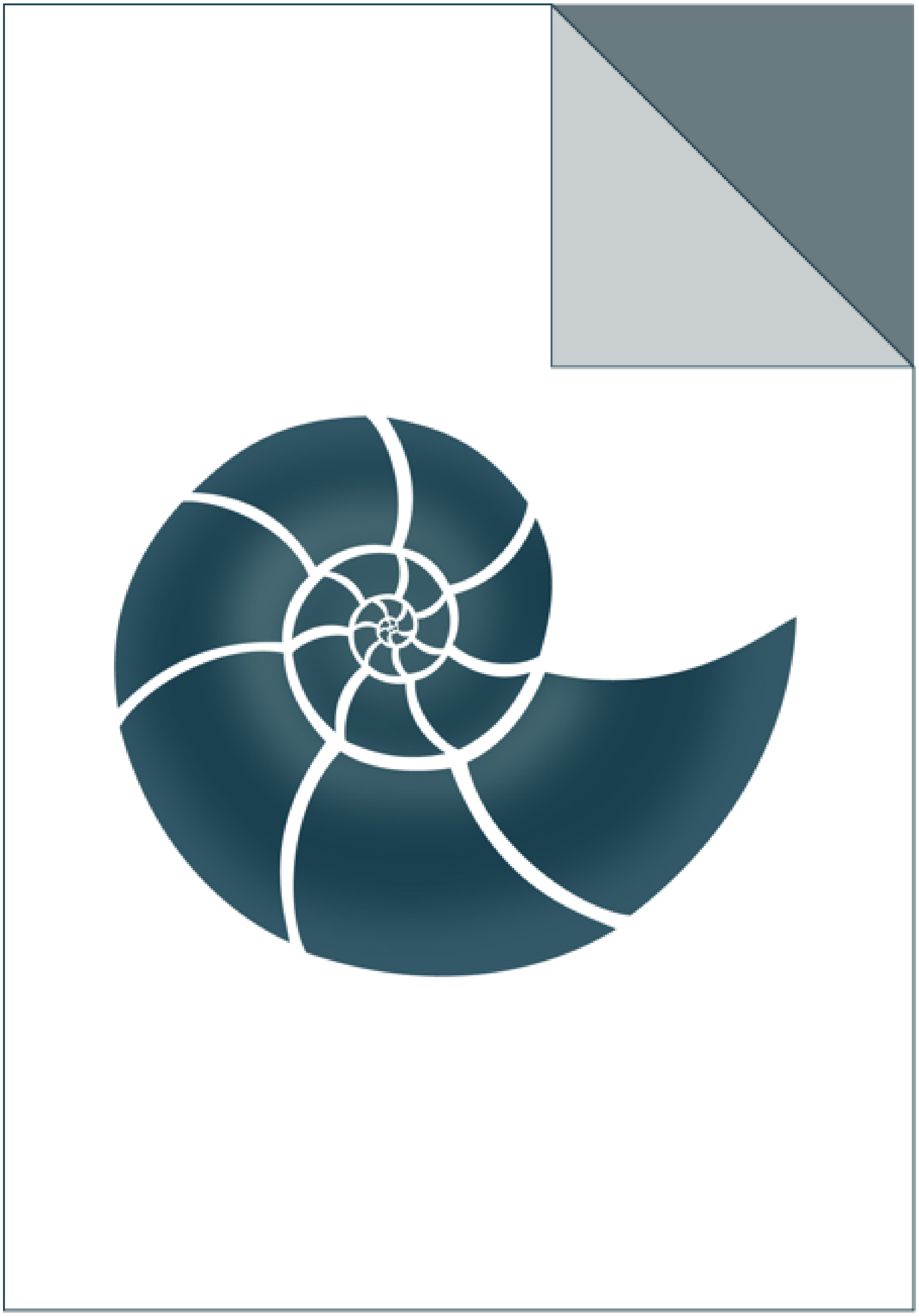
The program evaluates crmsd based on ligand cooridinates. It assumes the receptor structure doesn't change significantly and superimposes all models on the first one, which significantly reduces calculation time. If the reference structure is not given and dash '-' character is used instead (as in the last example), the program evaluates pairwise all-vs-all crmsd calculations.

Keywords:

- no_keywords

Categories:

- no_categories



5.1.47 ap_superimpose_pdb_by_ligand

Superimposes protein structures by matching ligand molecules.

All the given protein structures must contain the same ligand molecule, every time in the same conformation. The program calculates a transformation (rotation-translation) that superimposes that ligand from input structures on the same ligand molecule found in the native PDB. The transformation is then used to rototranslate whole protein structures. Results is written to "out.pdb" file

USAGE: ./ap_superimpose_pdb_by_ligand native_pdb ligand_name pdb_file_1 [pdb_file_2 . . .]

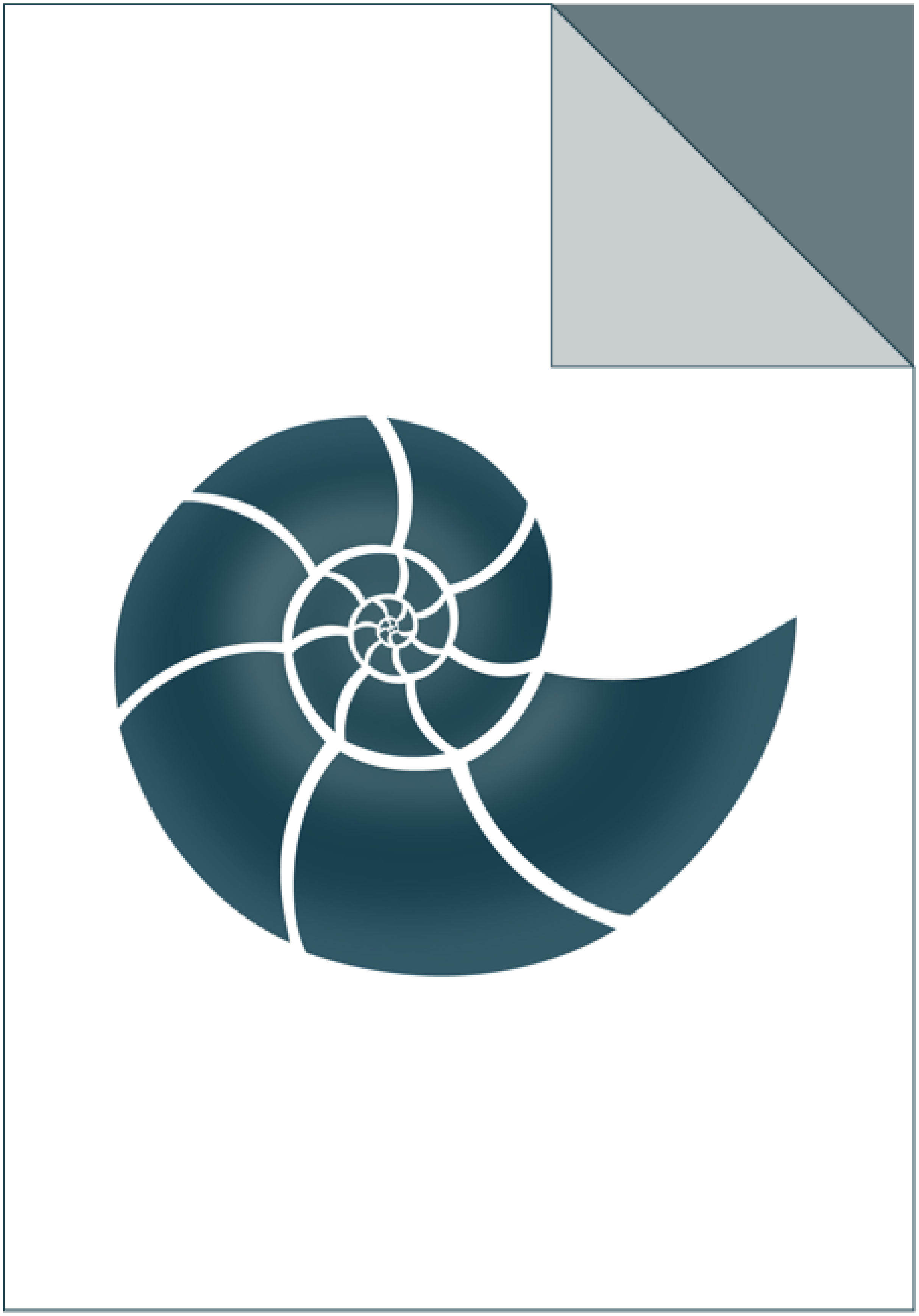
EXAMPLE: ./ap_superipose_pdb_by_ligand 4rm4A.pdb HEM 5ofqA.pd

Keywords:

- no_keywords

Categories:

- no_categories



5.2 *ex_** programs

These group contain unit test, i.e. programs that tests a single class of a function.

5.2.1 ex_BinaryTreeNode

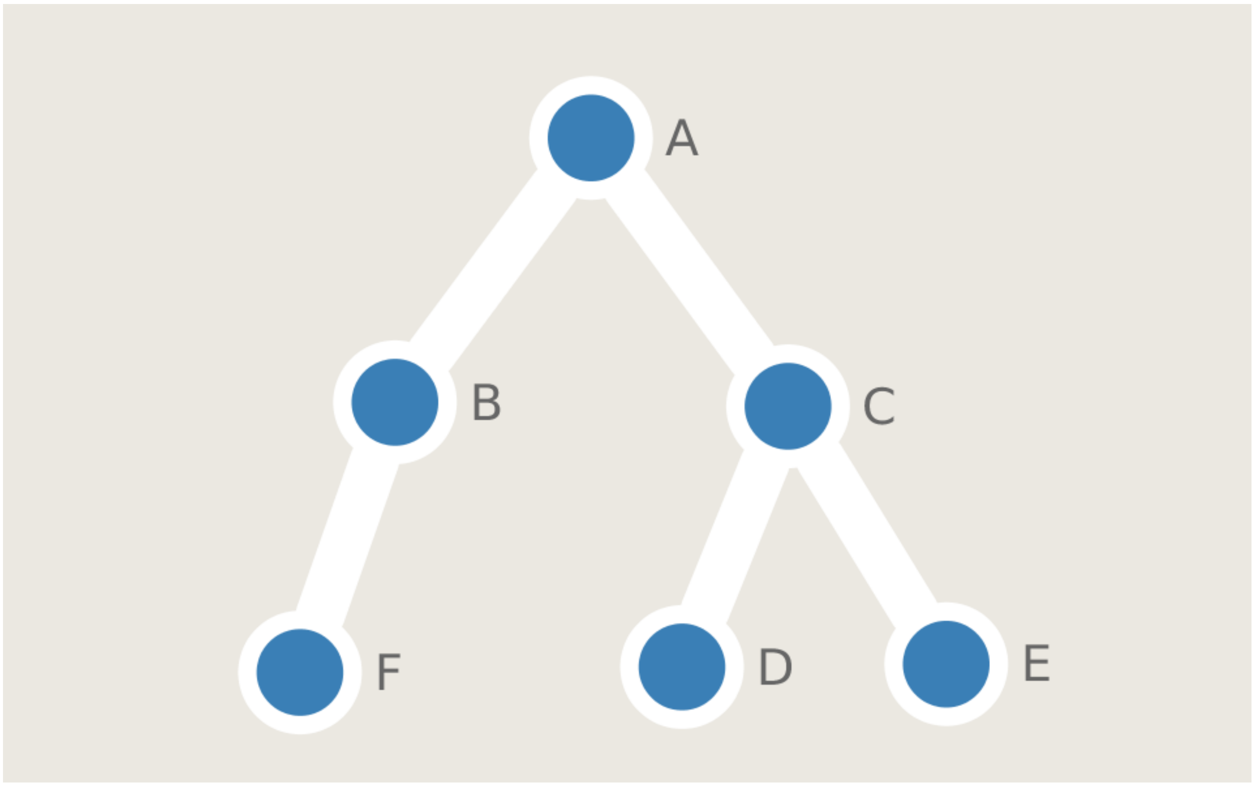
Simple demo for BinaryTreeNode class

Keywords:

- algorithms
- data structure
- depth first
- BinaryTreeNode

Categories:

- core/algorithms/trees/BinaryTreeNode



5.2.2 ex_Molecule

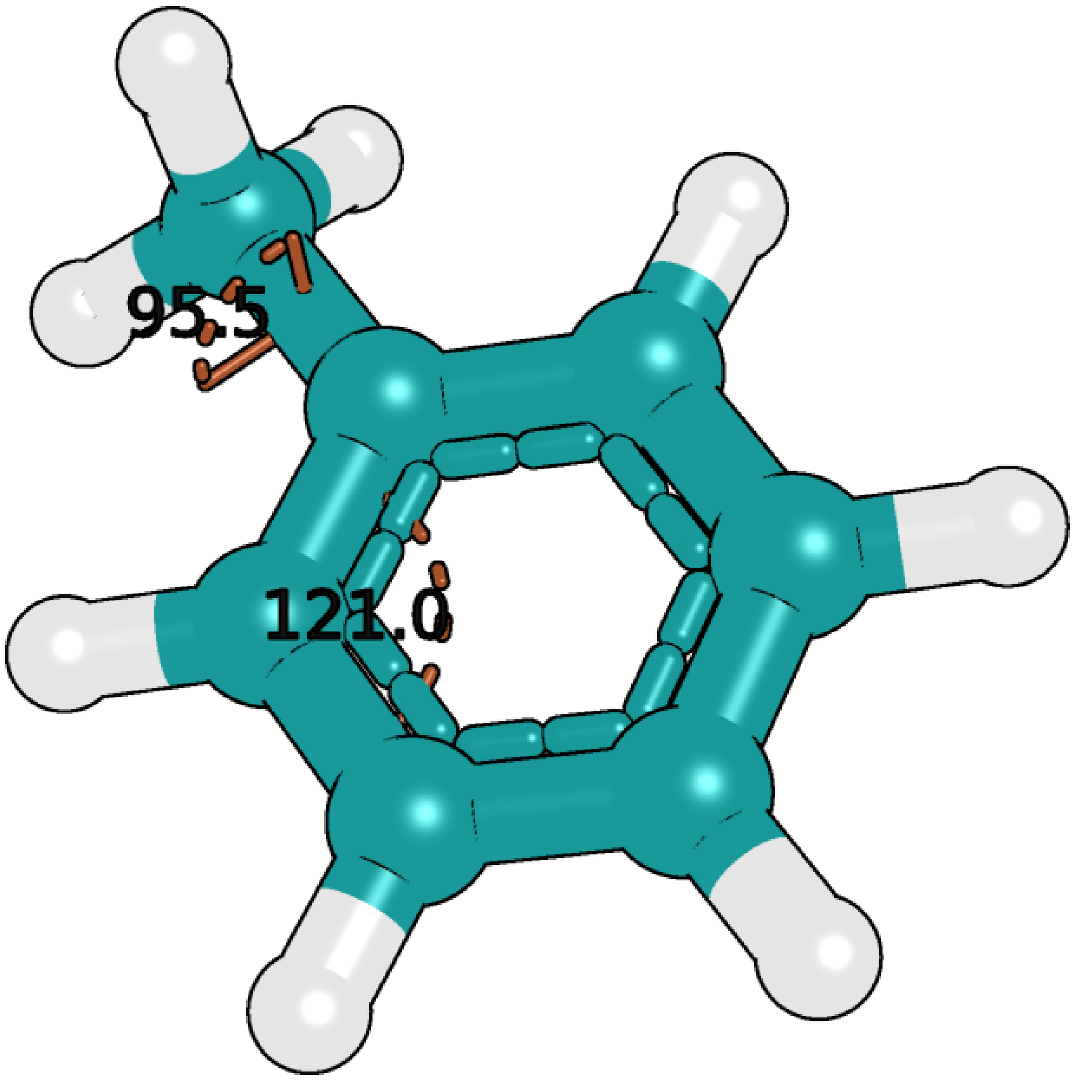
Demonstrates how to create a Molecule object based on PdbAtom data type (as nodes of the graph)

Keywords:

- molecule

Categories:

- core::chemical::Molecule



5.2.3 ex_Molecule_Vec3

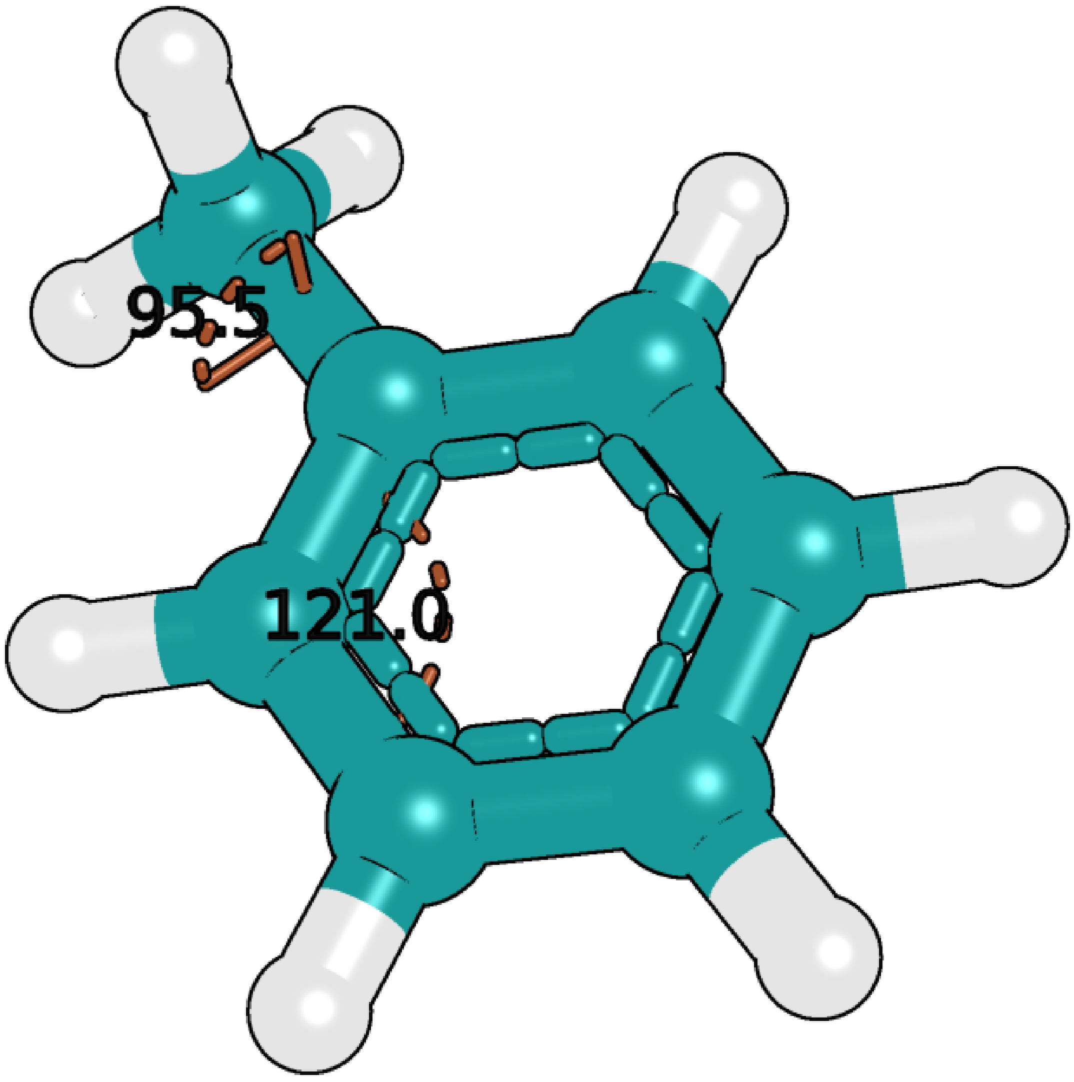
Demonstrates how to create a Molecule object based on Vec3 data type (Vec3 are nodes of the graph)

Keywords:

- molecule

Categories:

- core::chemical::Molecule



5.2.4 ex_NcbiSimilarityMatrixFactory

Test for loading substitution matrices available in BioShell.

The progam reads a substitution matrix (NCBI file format) and prints in back on the screen. If no input file is given, the program lists all the substitution matrices found in this BioShell distribution.

User can manually install custom matrices just by copying them to: data/alignments/ directory

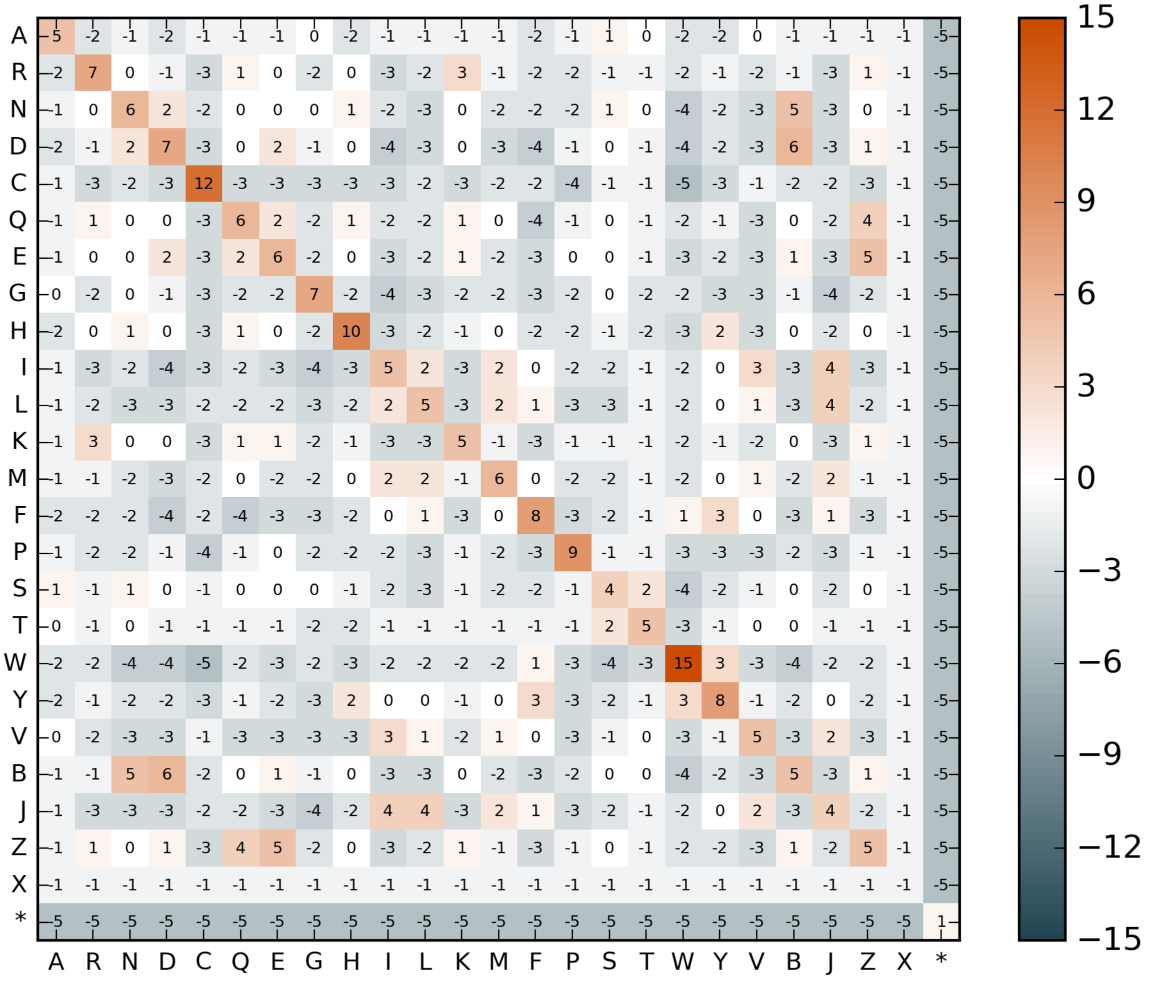
USAGE: ex_NcbiSimilarityMatrixFactory BLOSUM45.txt

Keywords:

- sequence alignment
- substitution matrix

Categories:

- core::alignment::scoring::NcbiSimilarityMatrixFactory



15
12
9
6
3
0
−3
−6
−9
−12
−15

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	J	Z	X	*
A	5	-2	-1	-2	-1	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-2	-2	0	-1	-1	-1	-1	-5
R	-2	7	0	-1	-3	1	0	-2	0	-3	-2	3	-1	-2	-2	-1	-1	-2	-1	-2	-1	-3	1	-1	-5
N	-1	0	6	2	-2	0	0	0	1	-2	-3	0	-2	-2	-2	1	0	-4	-2	-3	5	-3	0	-1	-5
D	-2	-1	2	7	-3	0	2	-1	0	-4	-3	0	-3	-4	-1	0	-1	-4	-2	-3	6	-3	1	-1	-5
C	-1	-3	-2	-3	12	-3	-3	-3	-3	-3	-2	-3	-2	-2	-4	-1	-1	-5	-3	-1	-2	-2	-3	-1	-5
Q	-1	1	0	0	-3	6	2	-2	1	-2	-2	1	0	-4	-1	0	-1	-2	-1	-3	0	-2	4	-1	-5
E	-1	0	0	2	-3	2	6	-2	0	-3	-2	1	-2	-3	0	0	-1	-3	-2	-3	1	-3	5	-1	-5
G	0	-2	0	-1	-3	-2	-2	7	-2	-4	-3	-2	-2	-3	-2	0	-2	-2	-3	-3	-1	-4	-2	-1	-5
H	-2	0	1	0	-3	1	0	-2	10	-3	-2	-1	0	-2	-2	-1	-2	-3	2	-3	0	-2	0	-1	-5
I	-1	-3	-2	-4	-3	-2	-3	-4	-3	5	2	-3	2	0	-2	-2	-1	-2	0	3	-3	4	-3	-1	-5
L	-1	-2	-3	-3	-2	-2	-2	-3	-2	2	5	-3	2	1	-3	-3	-1	-2	0	1	-3	4	-2	-1	-5
K	-1	3	0	0	-3	1	1	-2	-1	-3	-3	5	-1	-3	-1	-1	-1	-2	-1	-2	0	-3	1	-1	-5
M	-1	-1	-2	-3	-2	0	-2	-2	0	2	2	-1	6	0	-2	-2	-1	-2	0	1	-2	2	-1	-1	-5
F	-2	-2	-2	-4	-2	-4	-3	-3	-2	0	1	-3	0	8	-3	-2	-1	1	3	0	-3	1	-3	-1	-5
P	-1	-2	-2	-1	-4	-1	0	-2	-2	-2	-3	-1	-2	-3	9	-1	-1	-3	-3	-3	-2	-3	-1	-1	-5
S	1	-1	1	0	-1	0	0	0	-1	-2	-3	-1	-2	-2	-1	4	2	-4	-2	-1	0	-2	0	-1	-5
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-1	-1	2	5	-3	-1	0	0	-1	-1	-1	-5
W	-2	-2	-4	-4	-5	-2	-3	-2	-3	-2	-2	-2	-2	1	-3	-4	-3	15	3	-3	-4	-2	-2	-1	-5
Y	-2	-1	-2	-2	-3	-1	-2	-3	2	0	0	-1	0	3	-3	-2	-1	3	8	-1	-2	0	-2	-1	-5
V	0	-2	-3	-3	-1	-3	-3	-3	-3	3	1	-2	1	0	-3	-1	0	-3	-1	5	-3	2	-3	-1	-5
B	-1	-1	5	6	-2	0	1	-1	0	-3	-3	0	-2	-3	-2	0	0	-4	-2	-3	5	-3	1	-1	-5
J	-1	-3	-3	-3	-2	-2	-3	-4	-2	4	4	-3	2	1	-3	-2	-1	-2	0	2	-3	4	-2	-1	-5
Z	-1	1	0	1	-3	4	5	-2	0	-3	-2	1	-1	-3	-1	0	-1	-2	-2	-3	1	-2	5	-1	-5
X	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-5
*	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	1

5.2.5 ex_SelectChainResidueAtom

Extracts a fragment of a PDB file by applying a SelectChainResidueAtom selector.

The selection string constists of chain code and residue range, separated by a colon, e.g.: - A:-1-10 - AB:

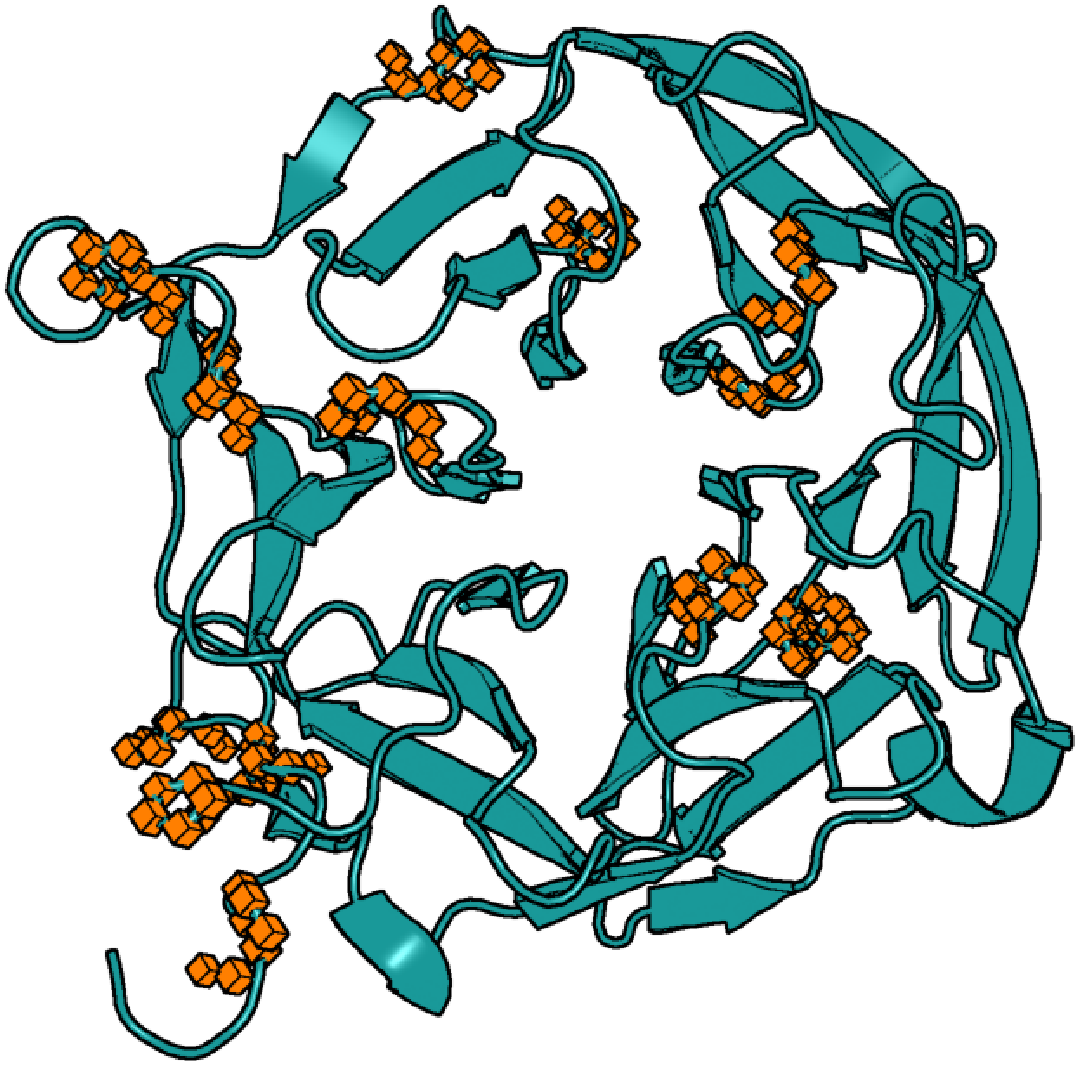
USAGE: ex_SelectChainResidueAtom test_inputs/2gb1.pdb A:23-32

Keywords:

- structure selectors
- PDB input
- PDB output

Categories:

- core::data::structural::StructureSelector



5.2.6 ex_SelectPlanarCAGeometry

> ex_SelectPlanarCAGeometry reads a PDB file and tests whether geometry at CA atom is tetrahedral or not.

The program also prints the actual values of the N-CA-C-CB dihedral angle. USAGE: ./ex_SelectPlanarCAGeometry 5edw.pdb

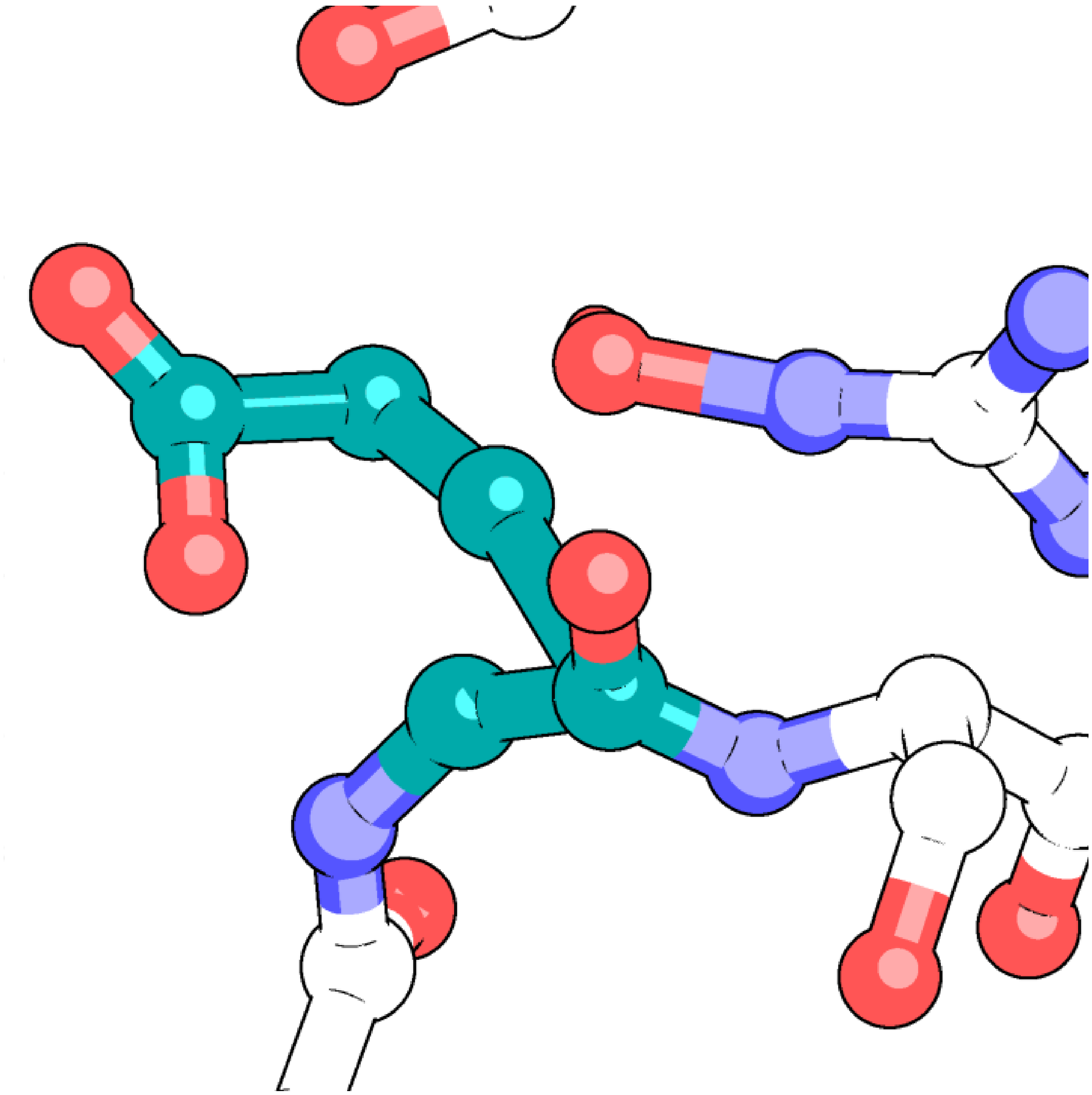
Keywords:

- residue geometry
- residue selectors

- PDB input
- structure validation

Categories:

- core::data::structural::ResidueHasBBCB; core::data::structural::SelectResidueByName; core::data::structural::SelectPlanarCAGeometry



5.2.7 ex_VonMisesDistribution

ex_VonMisesDistribution withdraws N random values (by default N = 1000) from a Normal distribution and fits ex_VonMises distribution to the data.

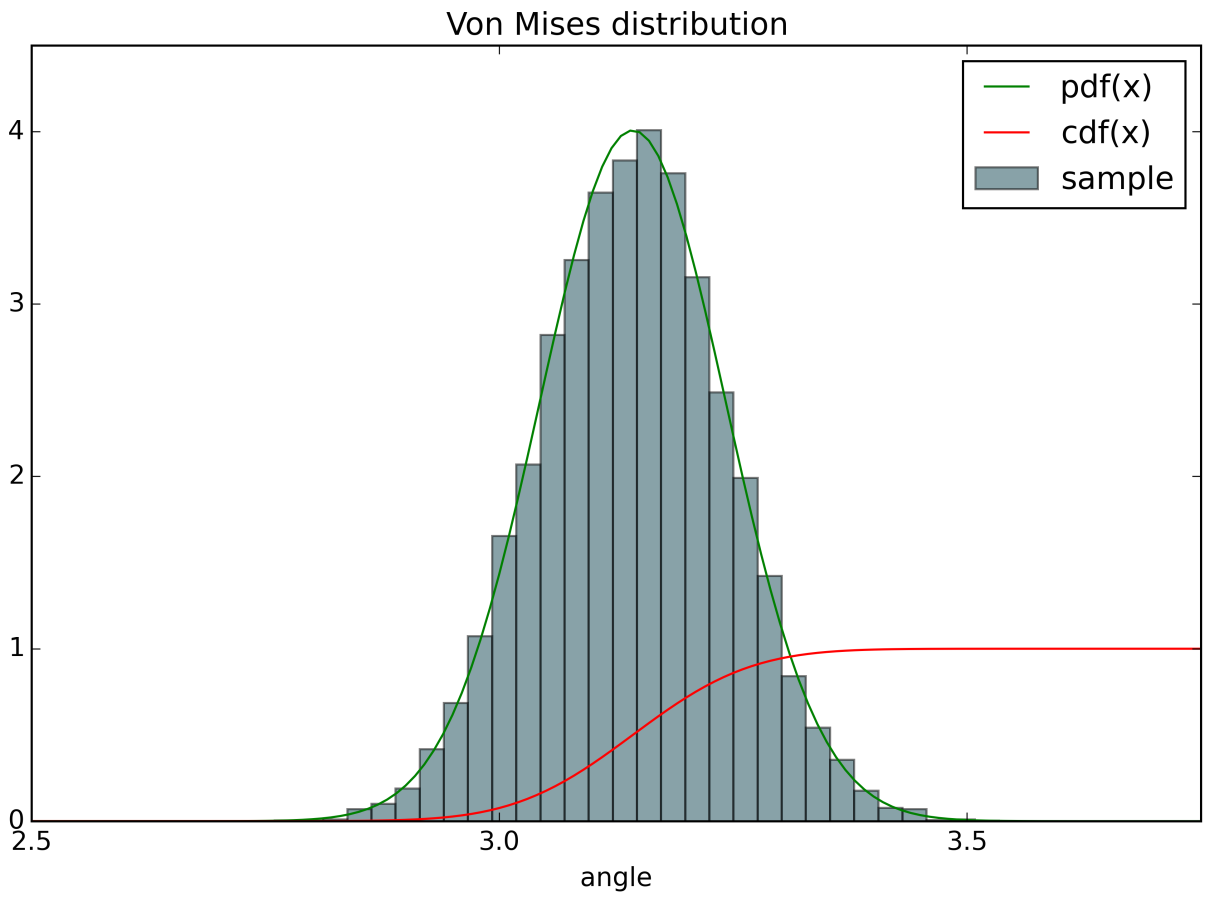
If exactly two arguments are provided (mu and kappa, respectively) the program tabulates Von Mises distribution for that parameters. USAGE: ex_VonMisesDistribution 10000 ex_VonMisesDistribution mu kappa

Keywords:

- statistics

Categories:

- core/calc/statistics/VonMisesDistribution



5.2.8 ex_bf_by_residue

ex_bf_by_residue reads a PDB file and prints per-residue statistics of B-factors. The output provides:

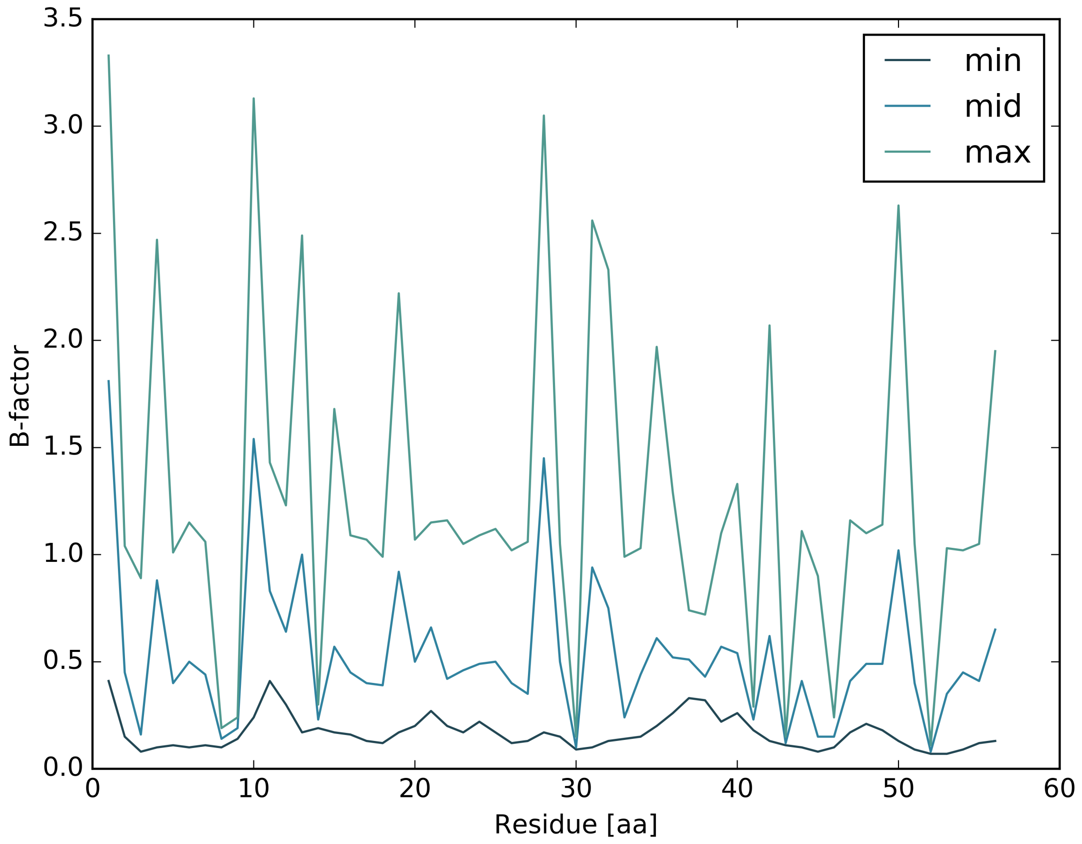
amino acid type (1-letter code), residue ID, and minimum, average and maximum b-factors for that residue USAGE: ex_bf_by_residue 2gb1.pdb

Keywords:

- PDB input
- B-factors
- atom selectors

Categories:

- core::data::io::Pdb



5.2.9 ex_chi_correlation

Calculates Chi dihedral angles of amino acid side chains measured in two different protein structures.

The program reads two homologous protein structures. The proteins are assumed to be already aligned and Chi angles are calculated for every pair of identical positions.

USAGE: ex_chi_correlation ./test_inputs/2fd2.pdb ./test_inputs/5fd1.pdb

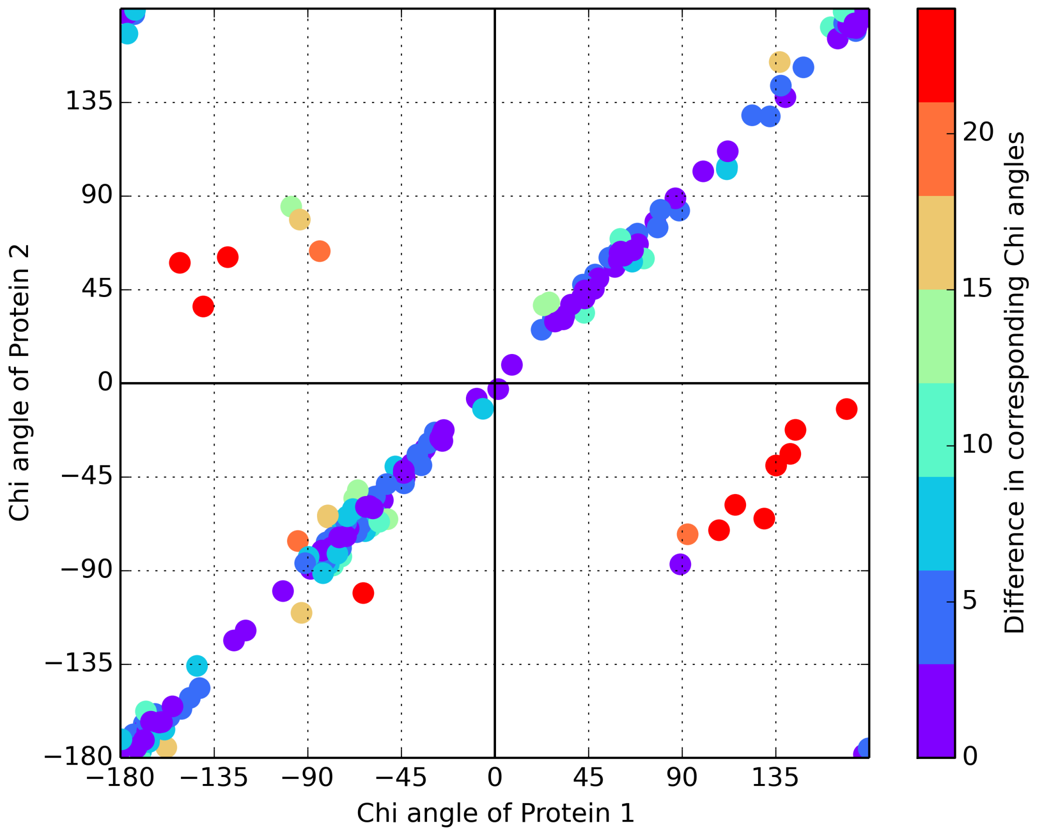
Keywords:

- PDB input

- chi angles
- rotamers

Categories:

- core/calc/structural/evaluate_chi



5.2.10 ex_evaluate_chi

Calculates all side chain Chi dihedral angles for the input protein structure

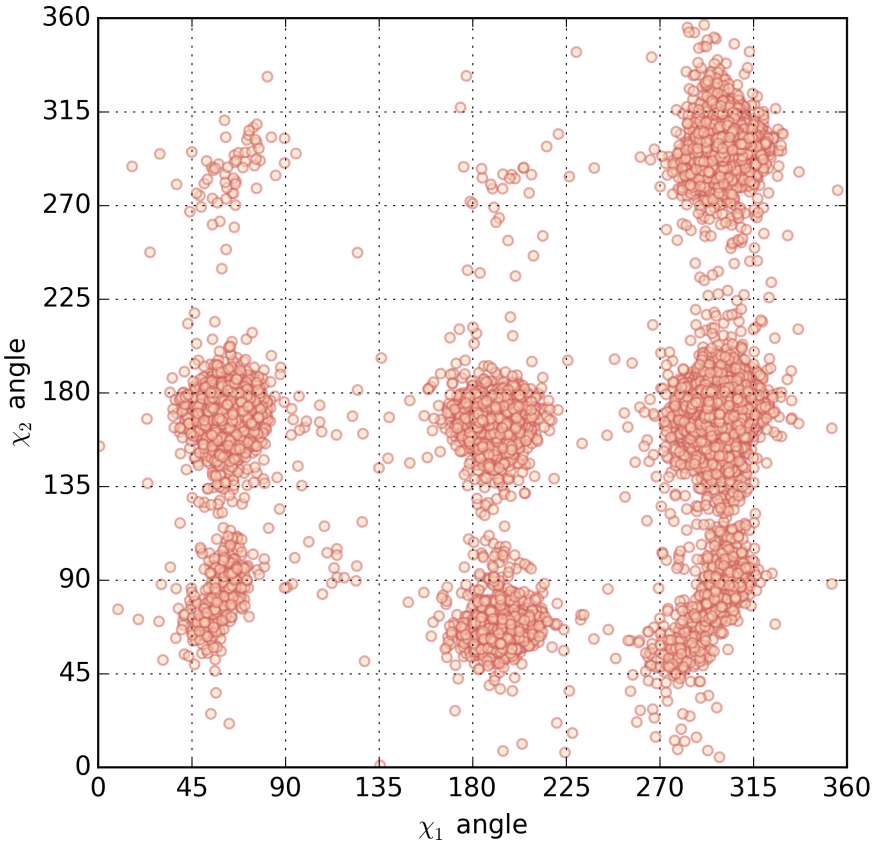
USAGE: ex_evaluate_chi 2kwi.pdb

Keywords:

- PDB input
- structure properties
- structure validation

Categories:

- core::chemical::ChiAnglesDefinition; core::calc::structural::evaluate_chi()



5.2.11 ex_evaluate_phi_psi

Calculates Phi,Psi angles (Ramachandran map) for every model found in the input protein structure

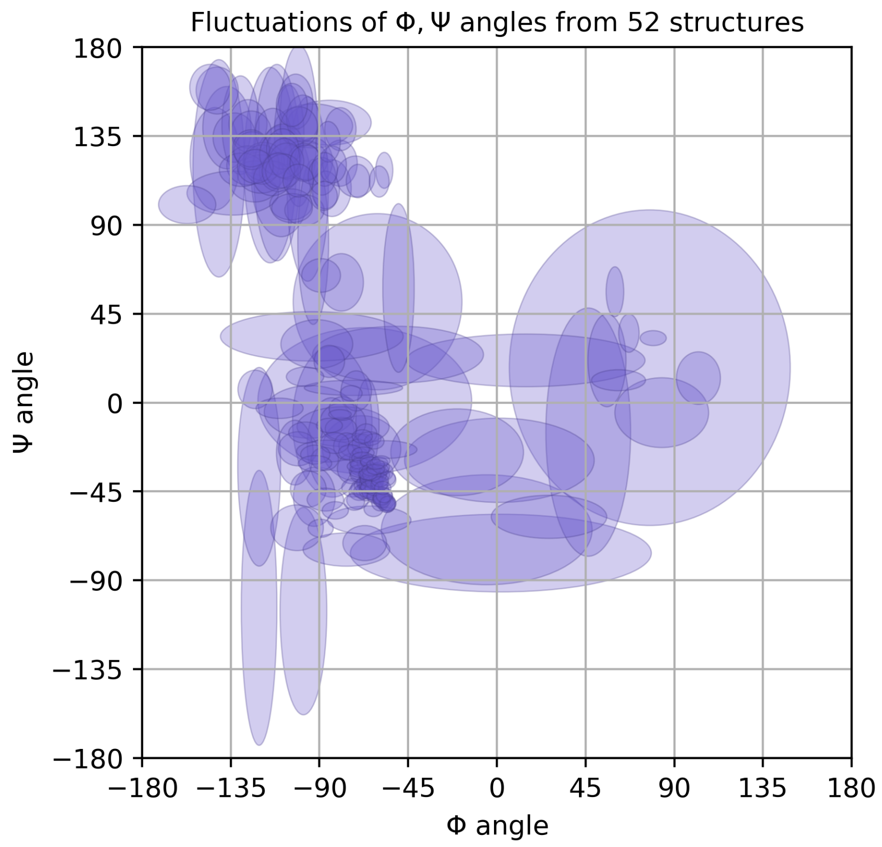
USAGE: ex_evaluate_phi_psi 2kwi.pdb

Keywords:

- PDB input
- structure properties
- structure validation
- Ramachandran map

Categories:

- core::calc::structural::LocalBackboneProperties



5.2.12 ex_plot_VonMises_mixture

ex_plot_VonMises_mixture evaluates a mixture of Von Mises distribution so it can be plotted nicely

USAGE: ex_plot_VonMises_mixture 0.487862 -3.00582 17.4059 0.0794212 -1.02886 112.164

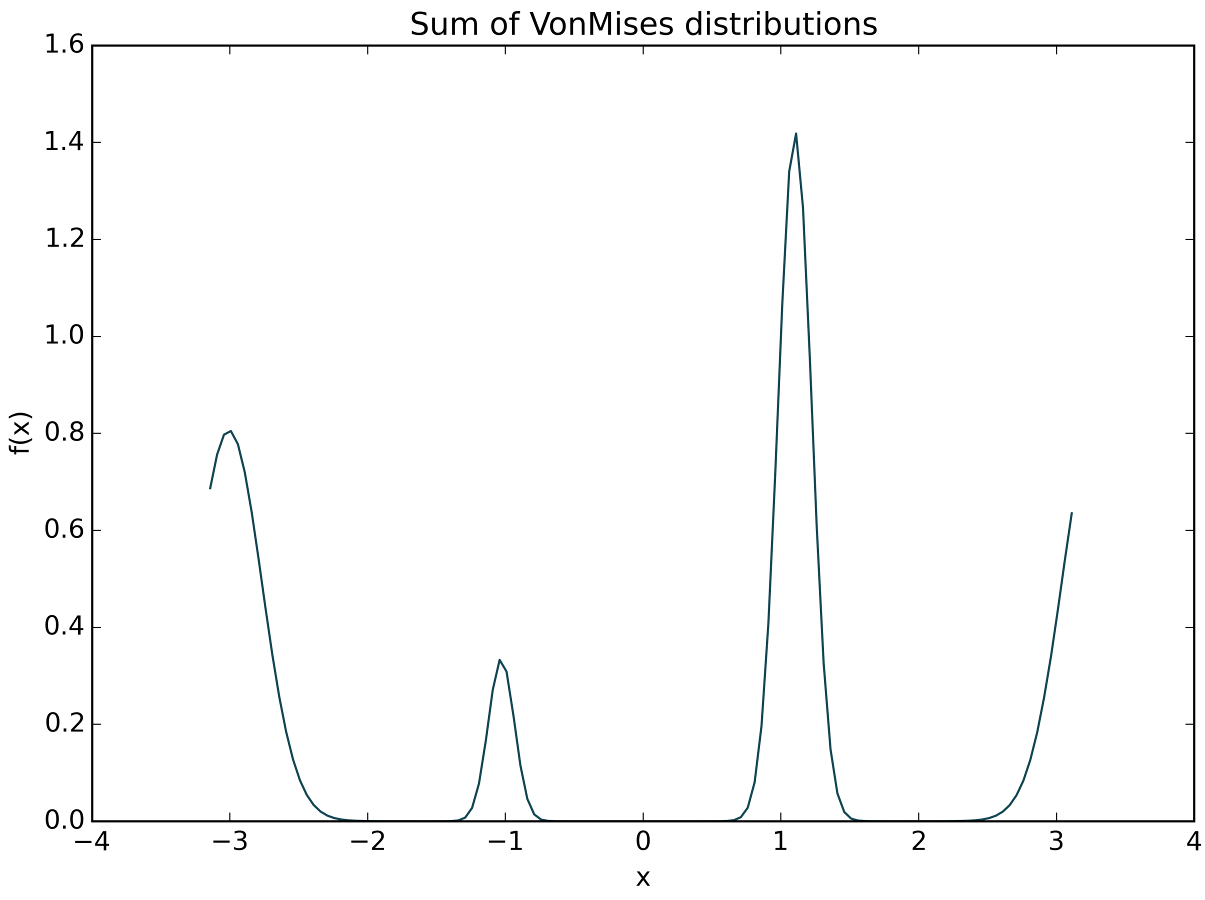
where the six numbers are scaling, mean and spread of two VonMises distribution

Keywords:

- statistics

Categories:

- core/calc/statistics/VonMisesDistribution



5.2.13 ex_Array2DSymmetric

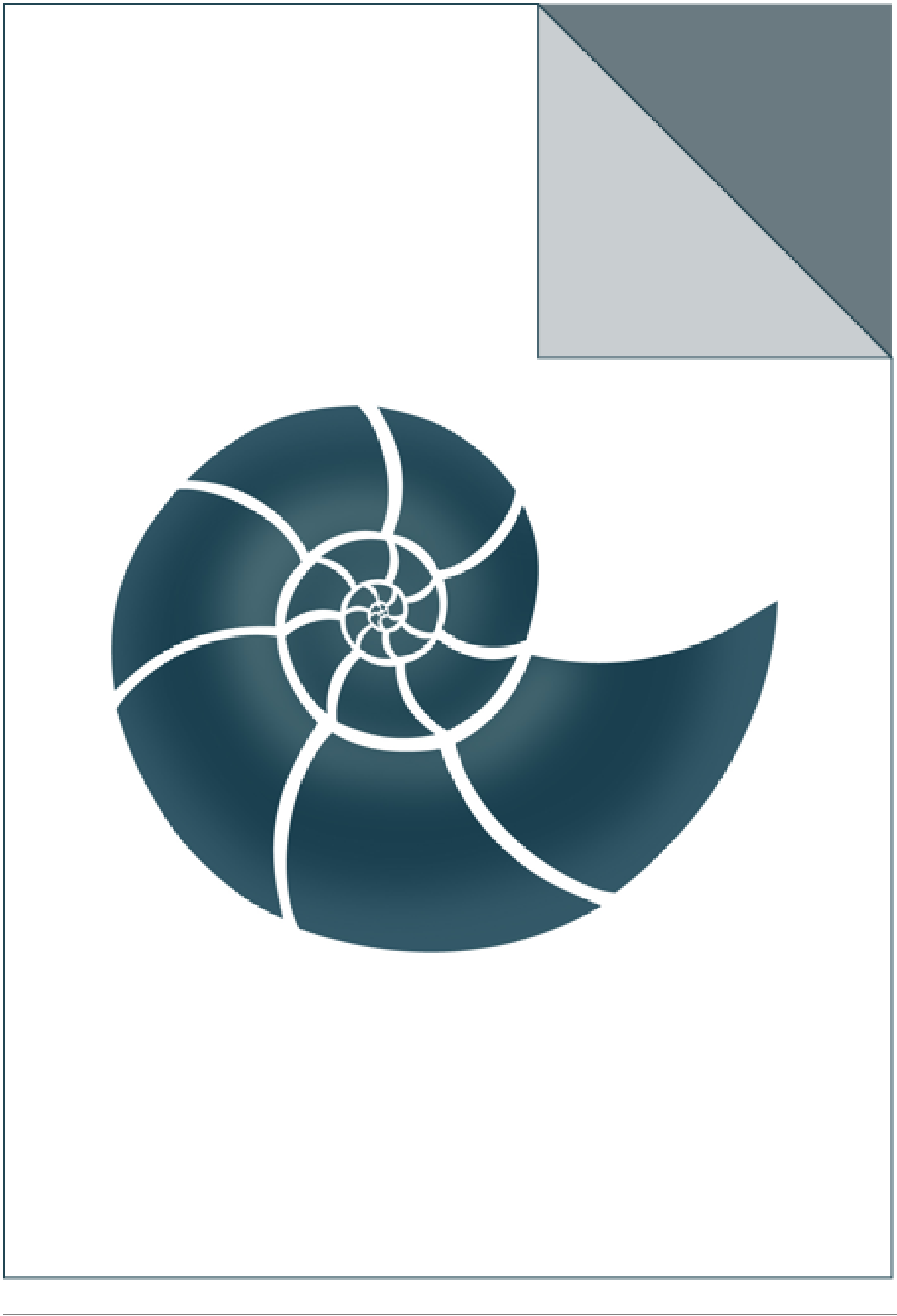
Simple test for Array2DSymmetric class.

Keywords:

- no_keywords

Categories:

- no_categories



5.2.14 ex_AtomSelector

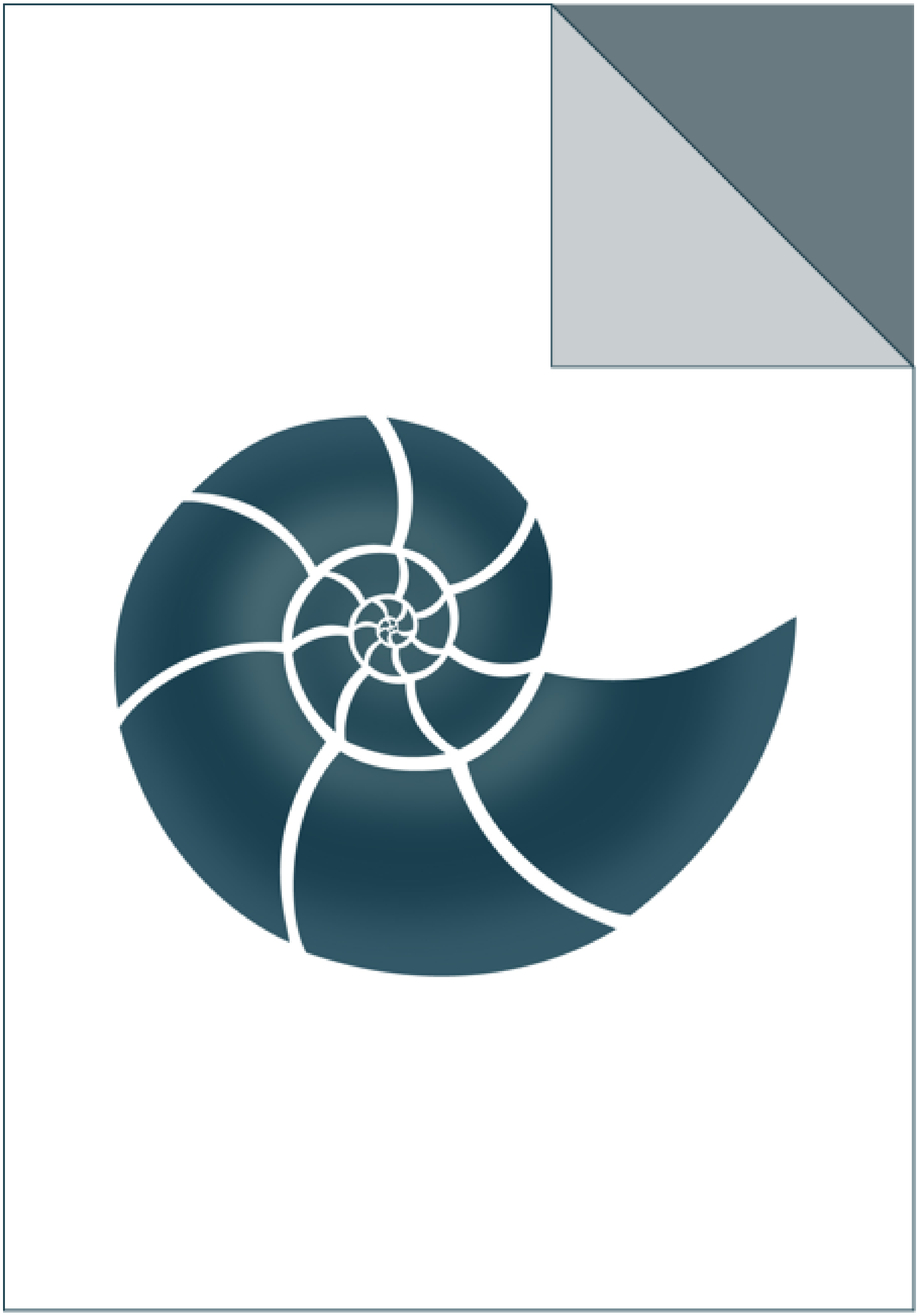
Demonstrates how to use atom selectors

Keywords:

- no_keywords

Categories:

- no_categories



5.2.15 ex_AtomicElement

Example showing how to use AtomicElement class

Keywords:

- no_keywords

Categories:

- no_categories

5.2.16 ex_BioShellVersion

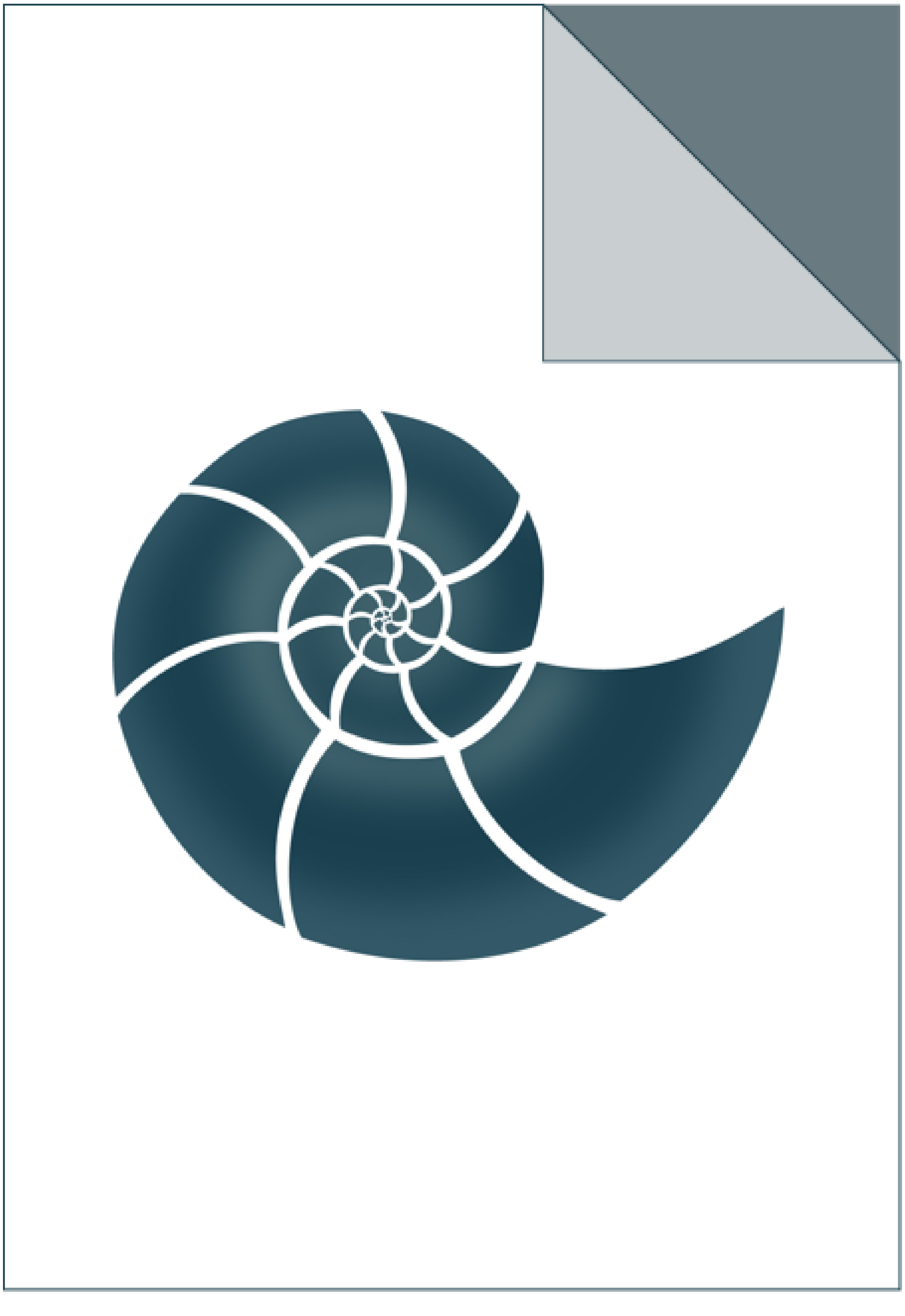
Test for BioShellVersion class prints the BioShell version info

Keywords:

- no_keywords

Categories:

- no_categories



5.2.17 ex_BitSet

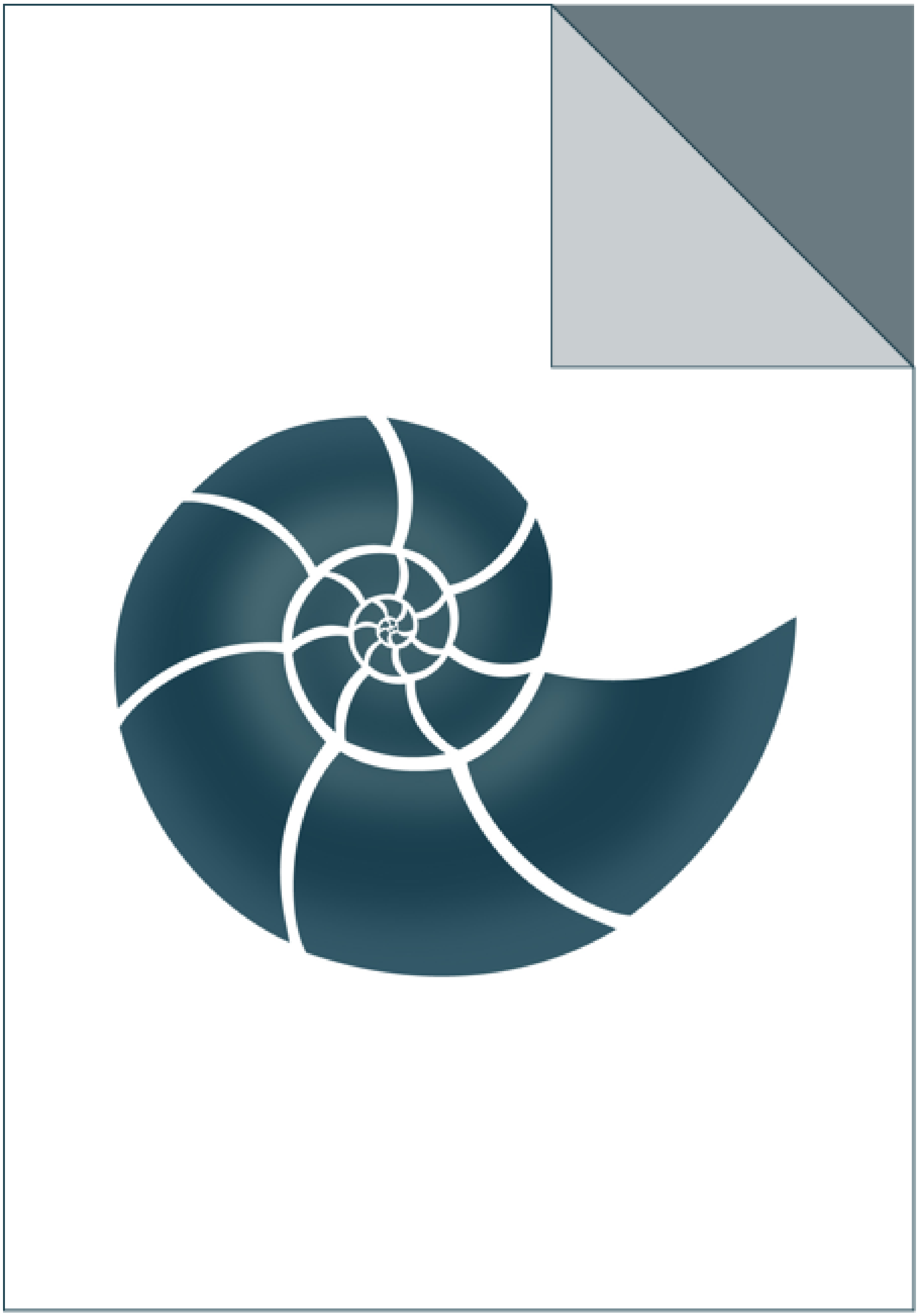
Test for BitSet class

Keywords:

- no_keywords

Categories:

- no_categories



5.2.18 ex_BivariateNormal

Estimates parameters of a two-dimensional Gaussian distribution

The program expects a file with columns of real values; based on them parameters of the distributions are estimated. Otherwise the example withdraws 10000 random numbers from a normal distribution and later it estimates a normal distribution from the sample.

USAGE: ex_BivariateNormal infile [x_column y_column]

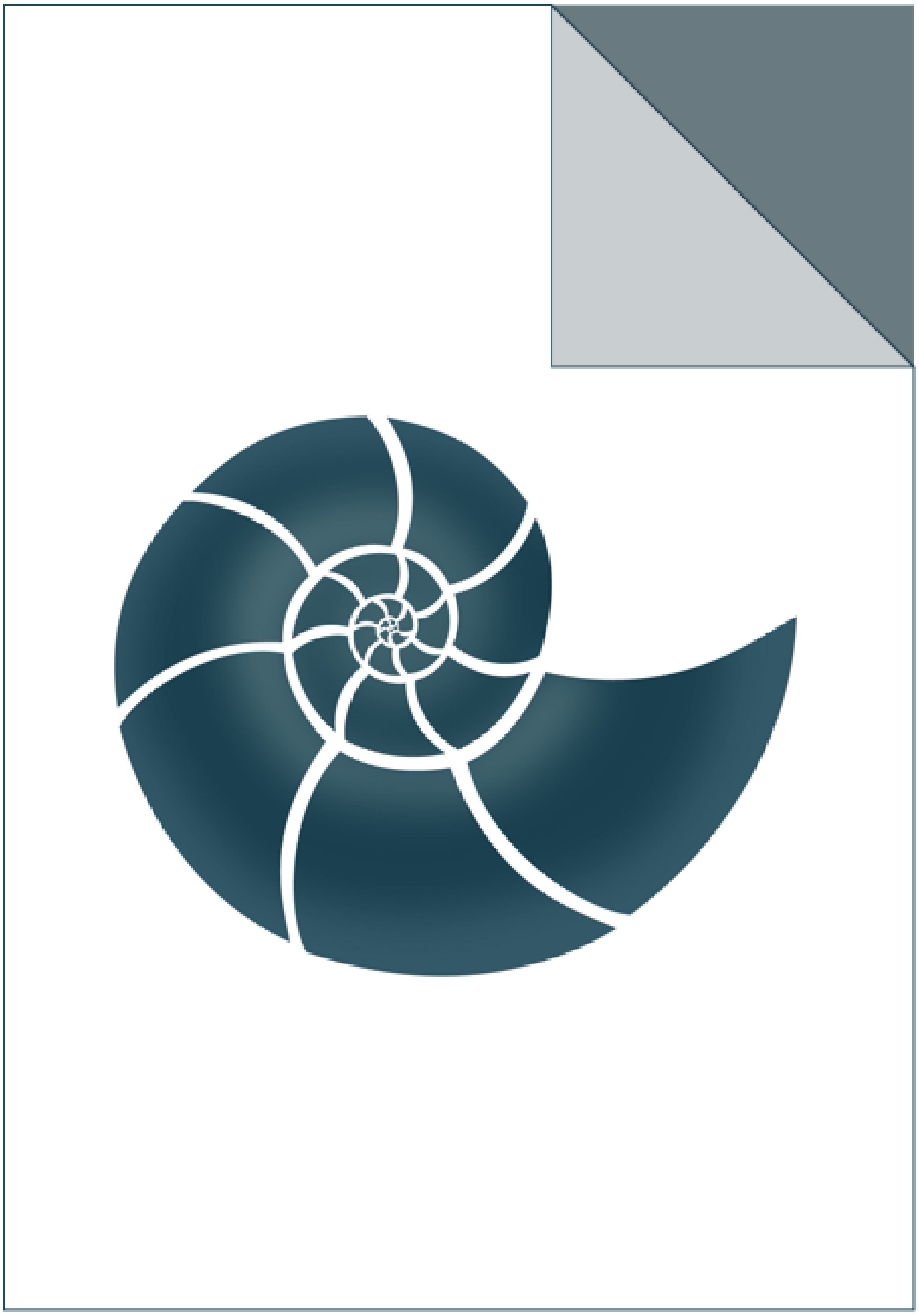
where x_column y_column are optional parameters tthat indicate which columns should be used for estimation; by default columns 0 and 1 are used.

Keywords:

- no_keywords

Categories:

- no_categories



5.2.19 ex_BoundedPriorityQueue

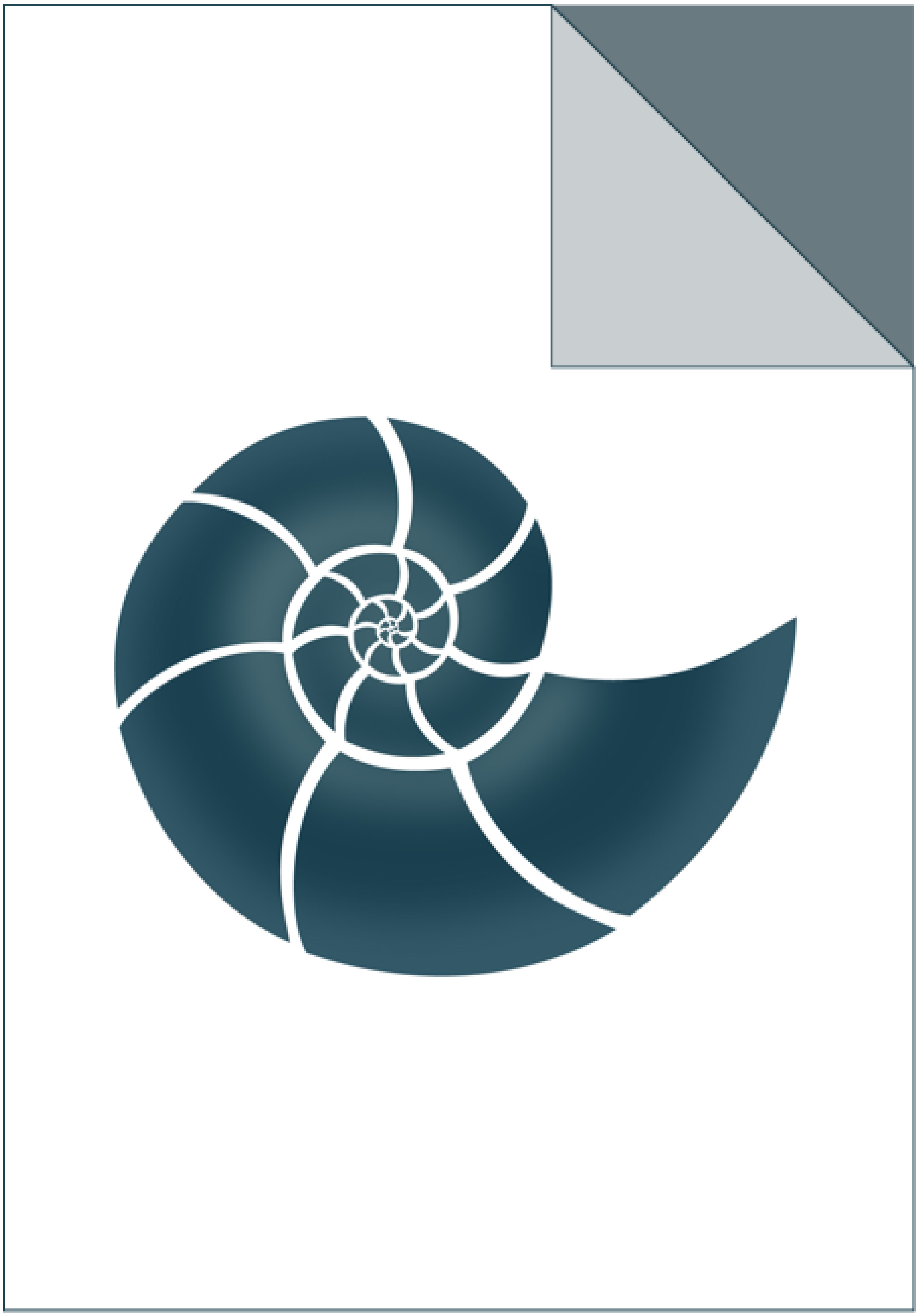
Simple demo for BoundedPriorityQueue class

Keywords:

- no_keywords

Categories:

- no_categories



5.2.20 ex_Cart

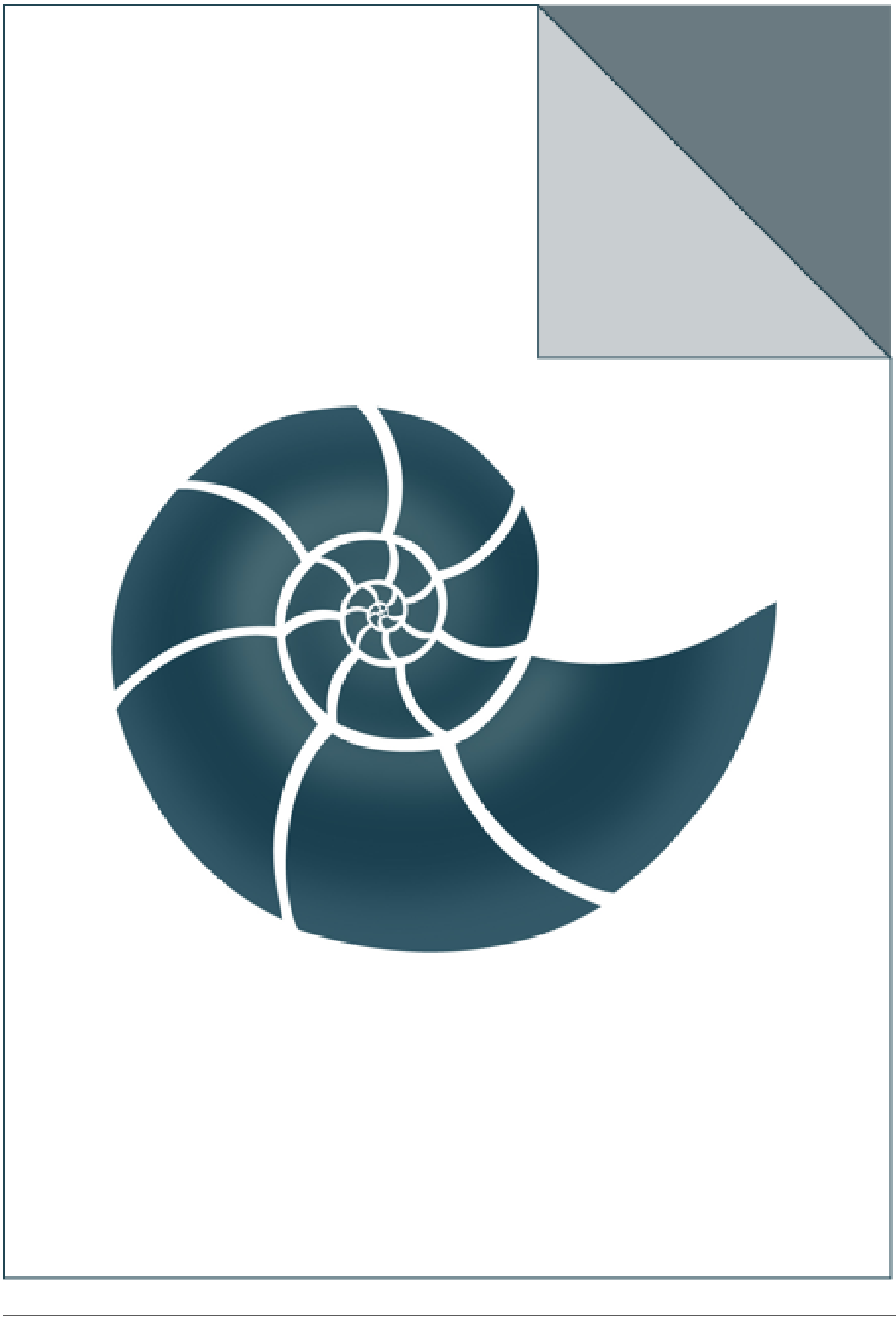
Shows how to use CART classification model

Keywords:

- no_keywords

Categories:

- no_categories



5.2.21 ex_CartesianToSpherical

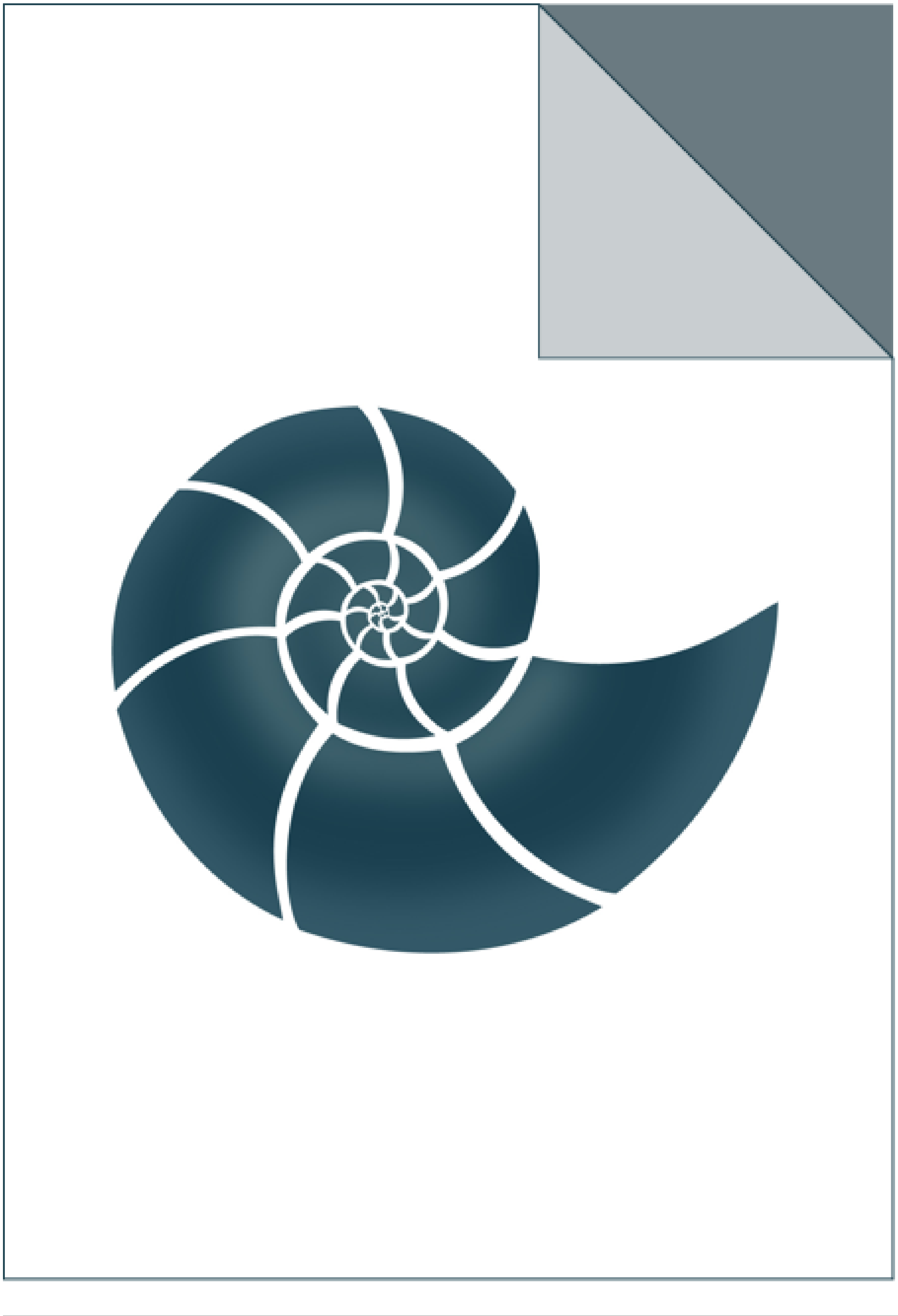
Calculates spherical coordinates using BioShell and ‘by hand’ to check of it works

Keywords:

- no_keywords

Categories:

- no_categories



5.2.22 ex_ChiAnglesDefinition

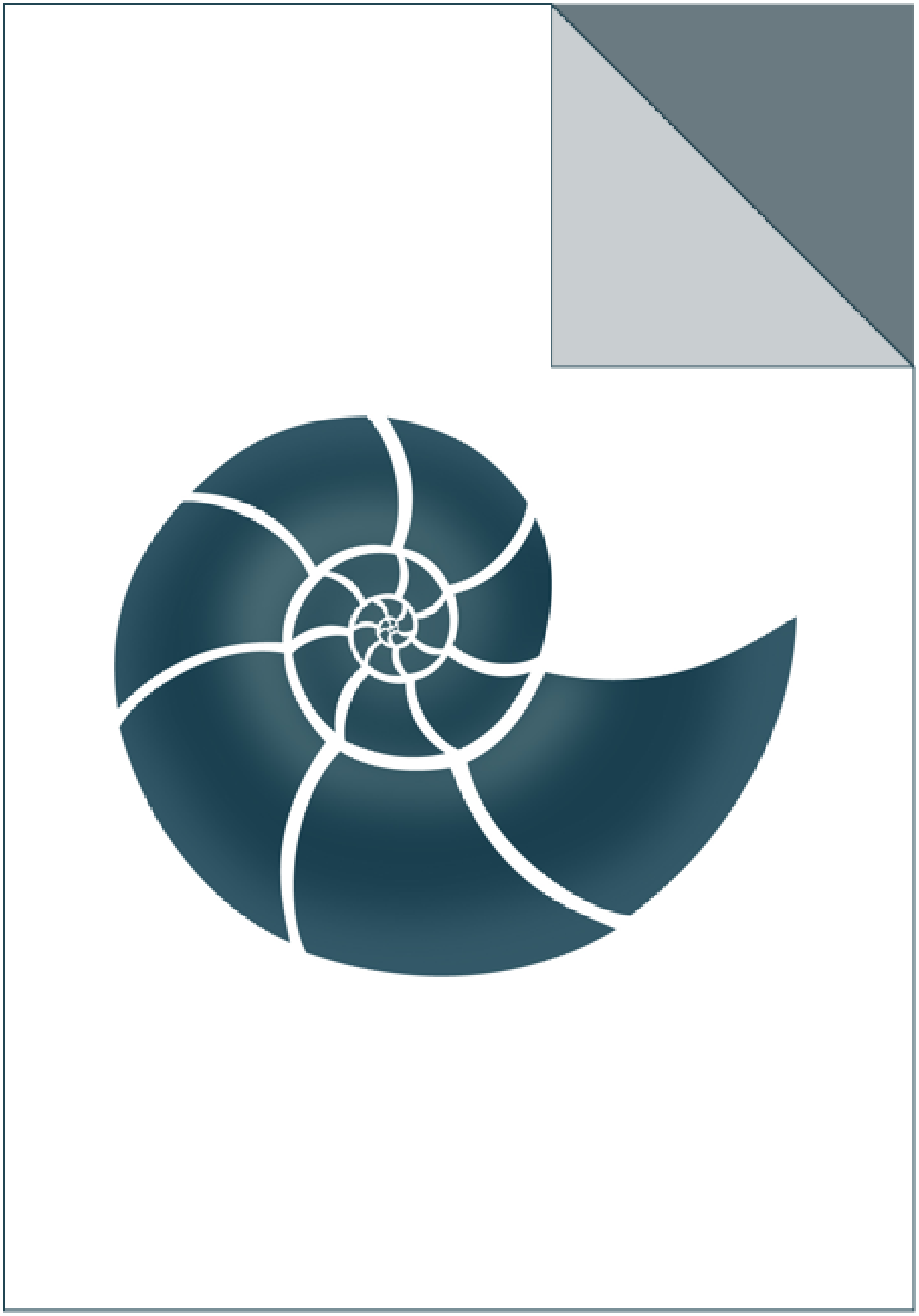
Shows how to look up information on Chi angle definitions

Keywords:

- no_keywords

Categories:

- no_categories



5.2.23 ex_Cif

ex_Cif tests reading CIF files

USAGE: ex_Cif file.cif

Keywords:

- no_keywords

Categories:

- no_categories

5.2.24 ex_Combinations

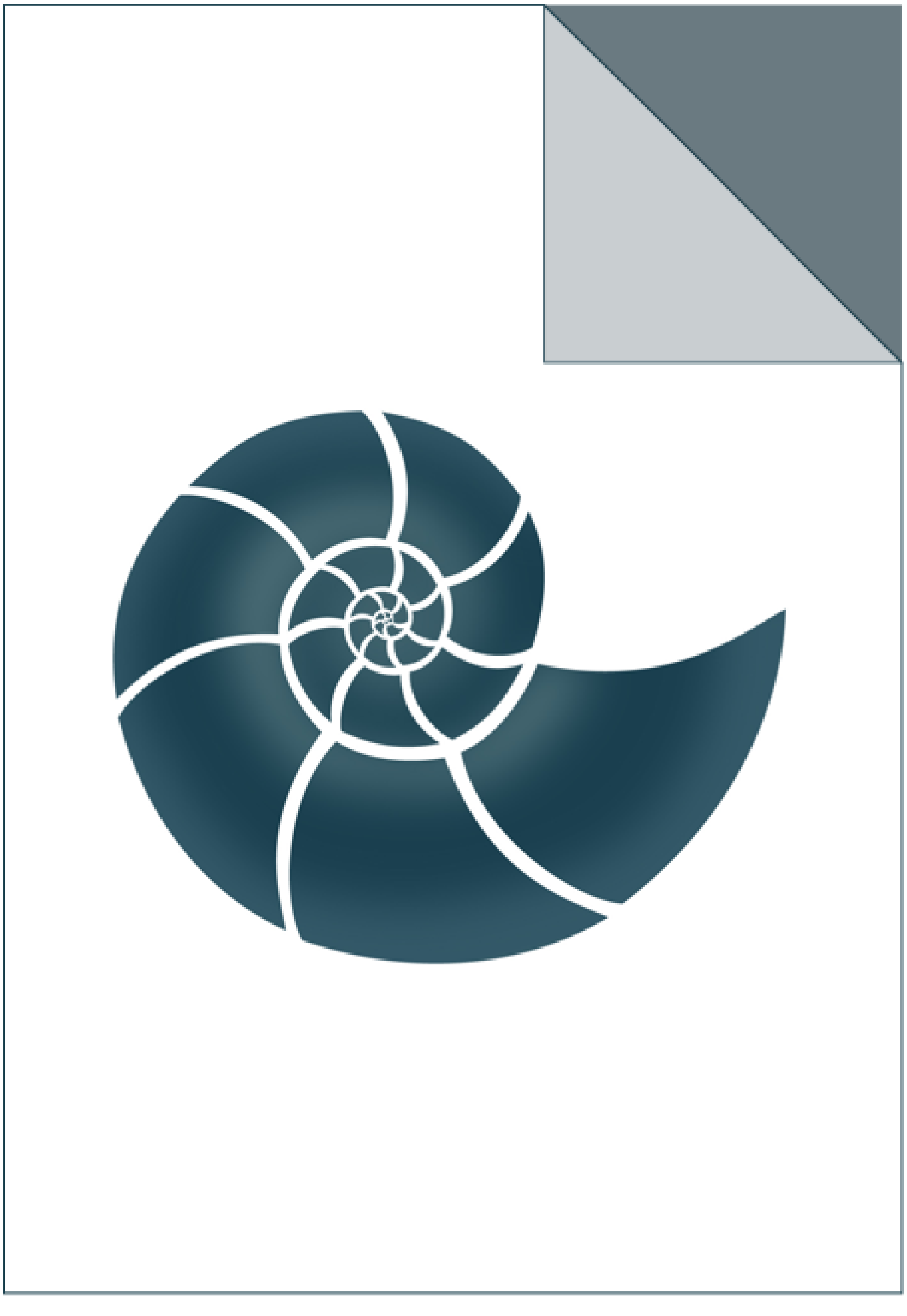
A simple example shows how to generate Combination

Keywords:

- no_keywords

Categories:

- no_categories



5.2.25 ex_DsspData

ex_DsspData reads a DSSP file and writes secondary structure in FASTA format

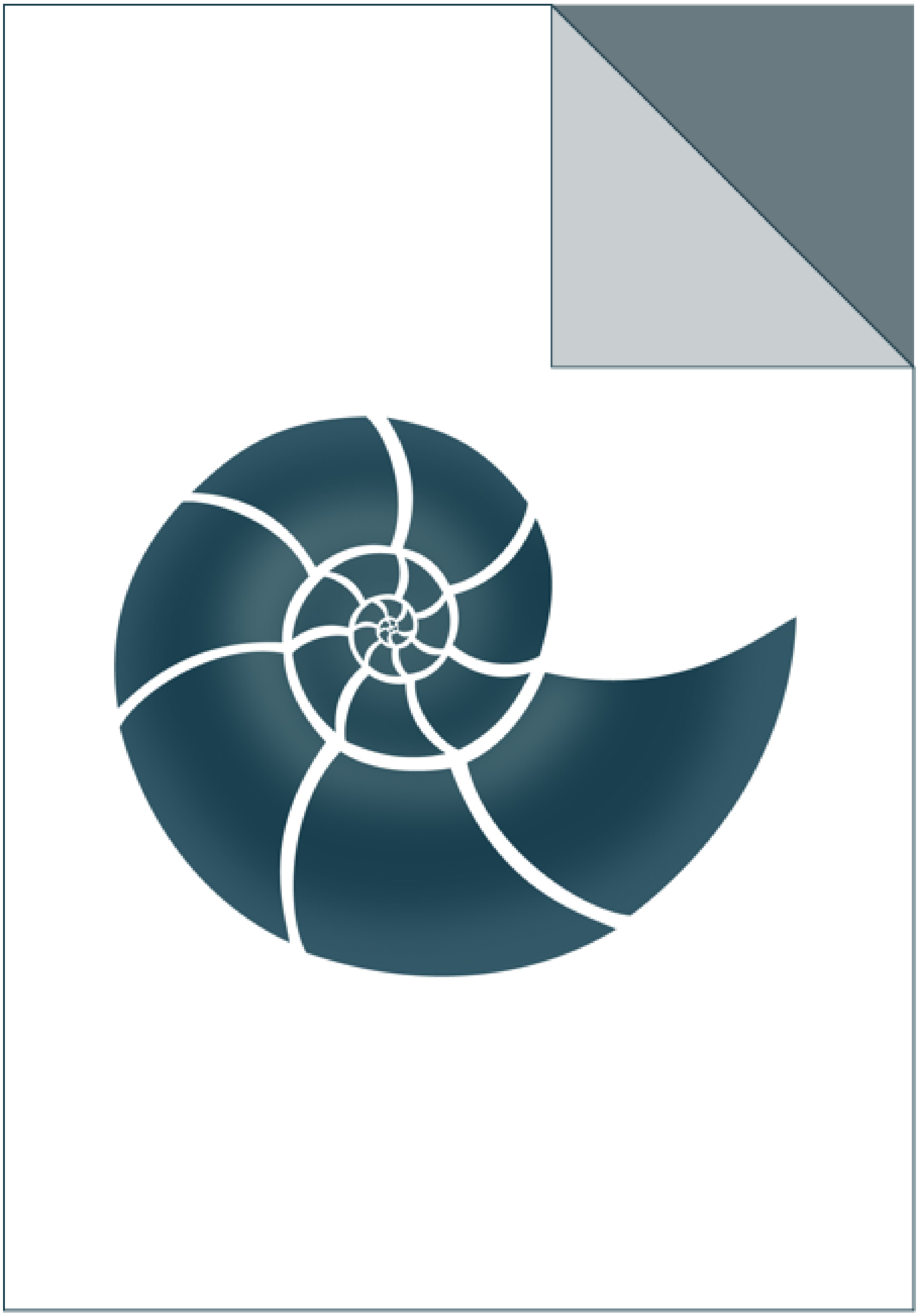
USAGE: ex_DsspData 5edw.dssp

Keywords:

- no_keywords

Categories:

- no_categories



5.2.26 ex_HierarchicalClustering

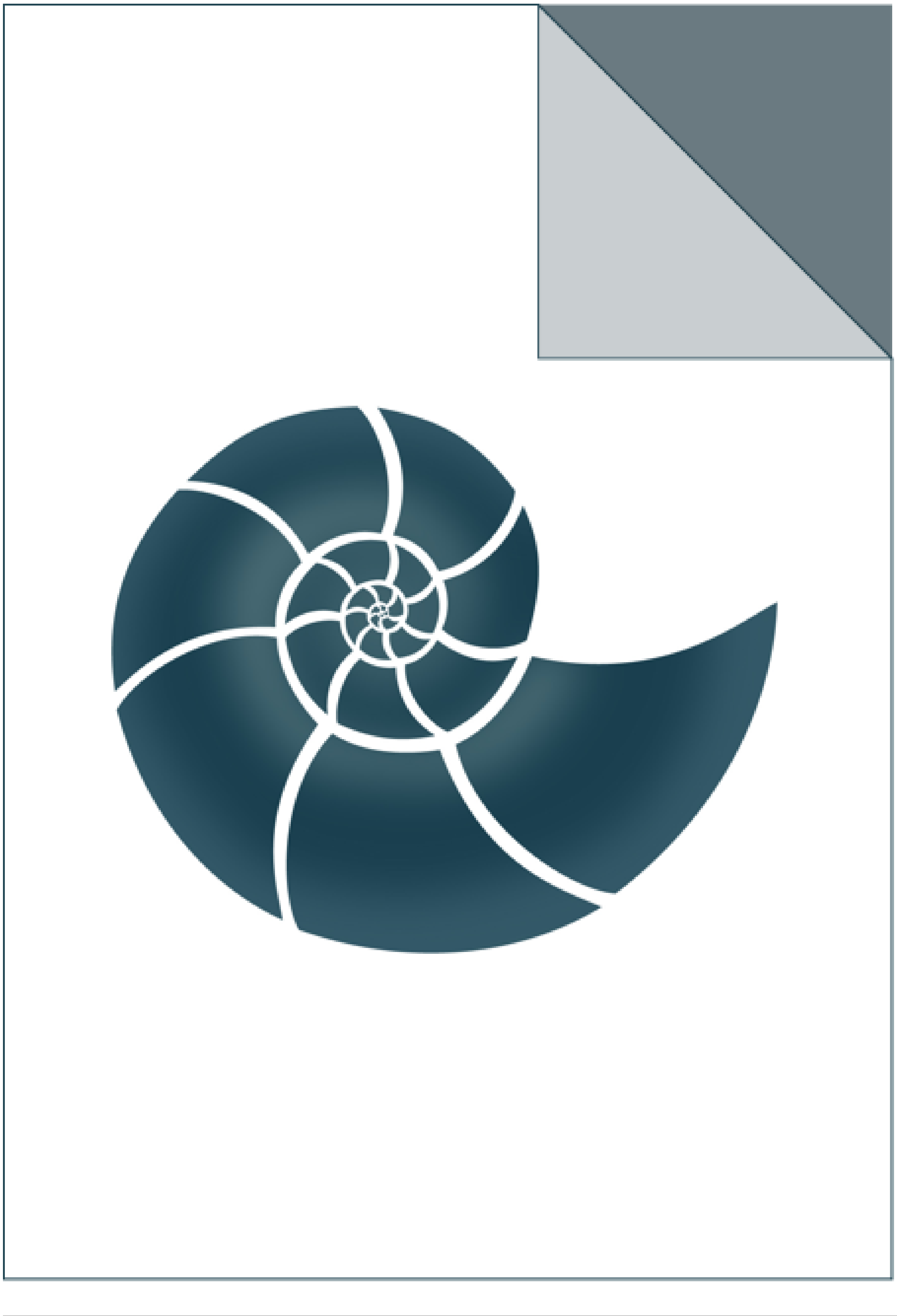
Example showing how to use hierarchical clustering method.

Keywords:

- no_keywords

Categories:

- no_categories



5.2.27 ex_Interpolate1D

ex_Interpolate1D reads a file with two columns of data and calculates interpolated values

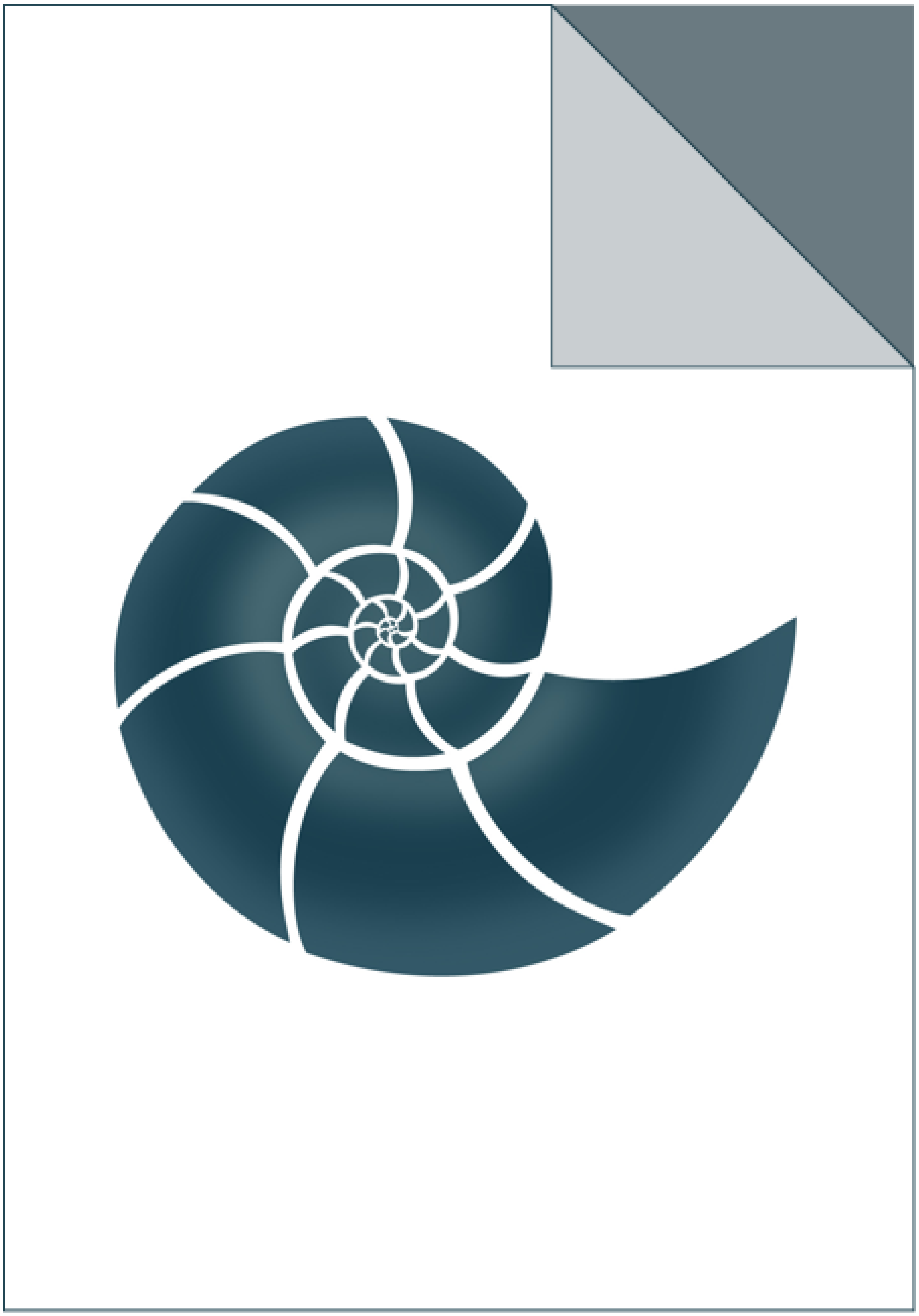
USAGE: ex_Interpolate1D infile n_steps

Keywords:

- no_keywords

Categories:

- no_categories



5.2.28 ex_InterpolatePeriodic1D

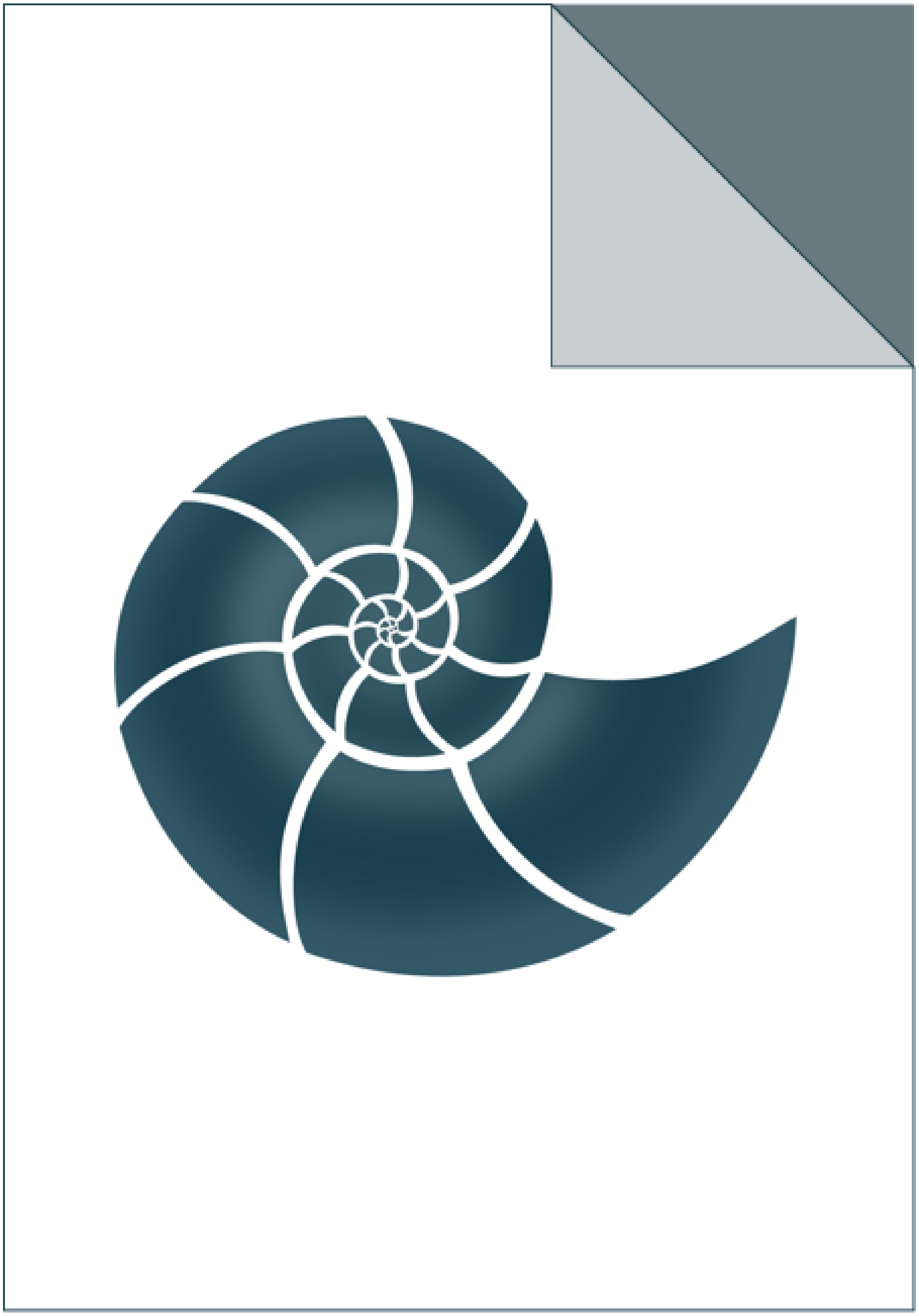
Simple test for interpolation of a periodic 1D function

Keywords:

- no_keywords

Categories:

- no_categories



5.2.29 ex_InterpolatePeriodic2D

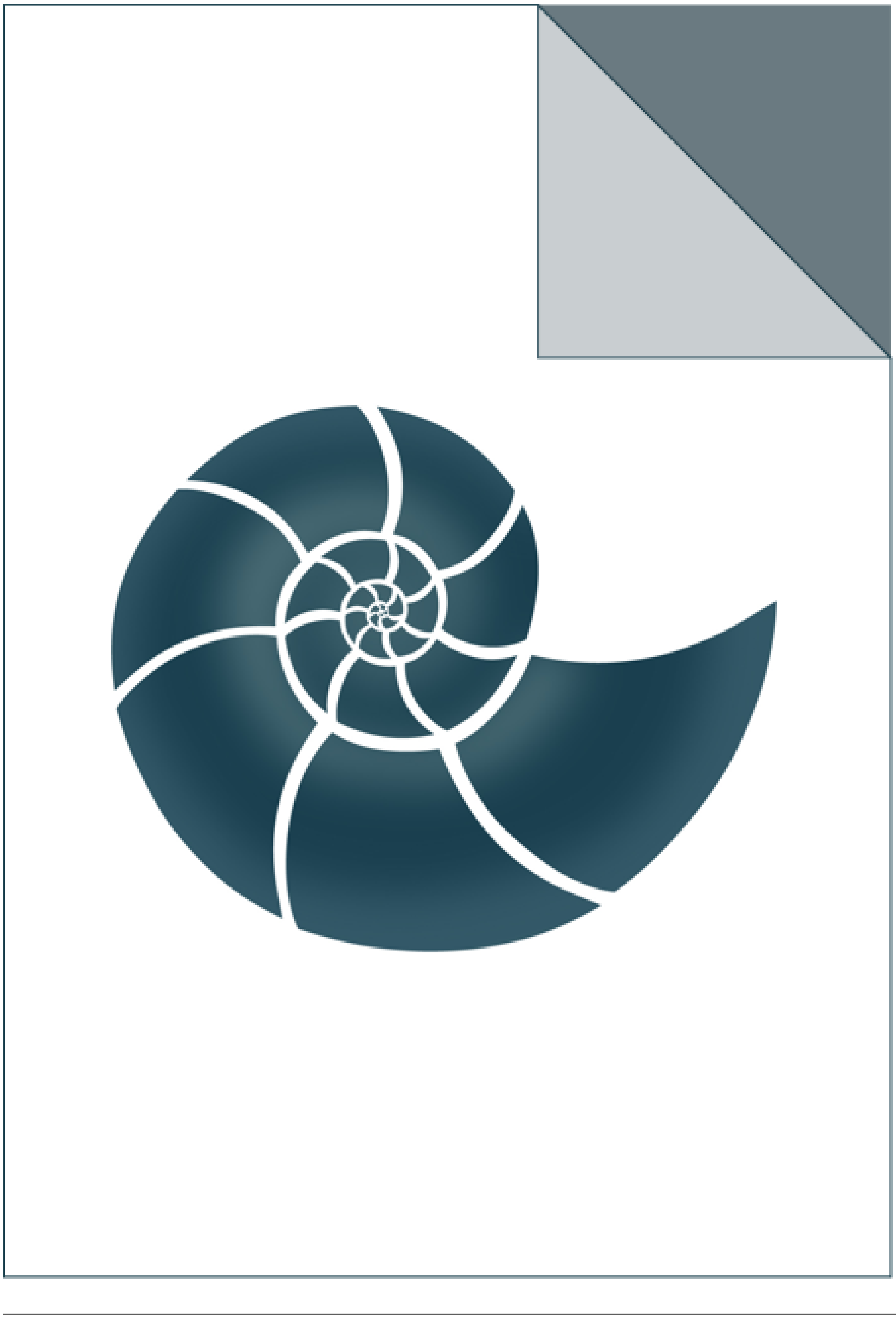
Simple test for interpolation of a periodic 2D function

Keywords:

- no_keywords

Categories:

- no_categories



5.2.30 ex_JsonNode

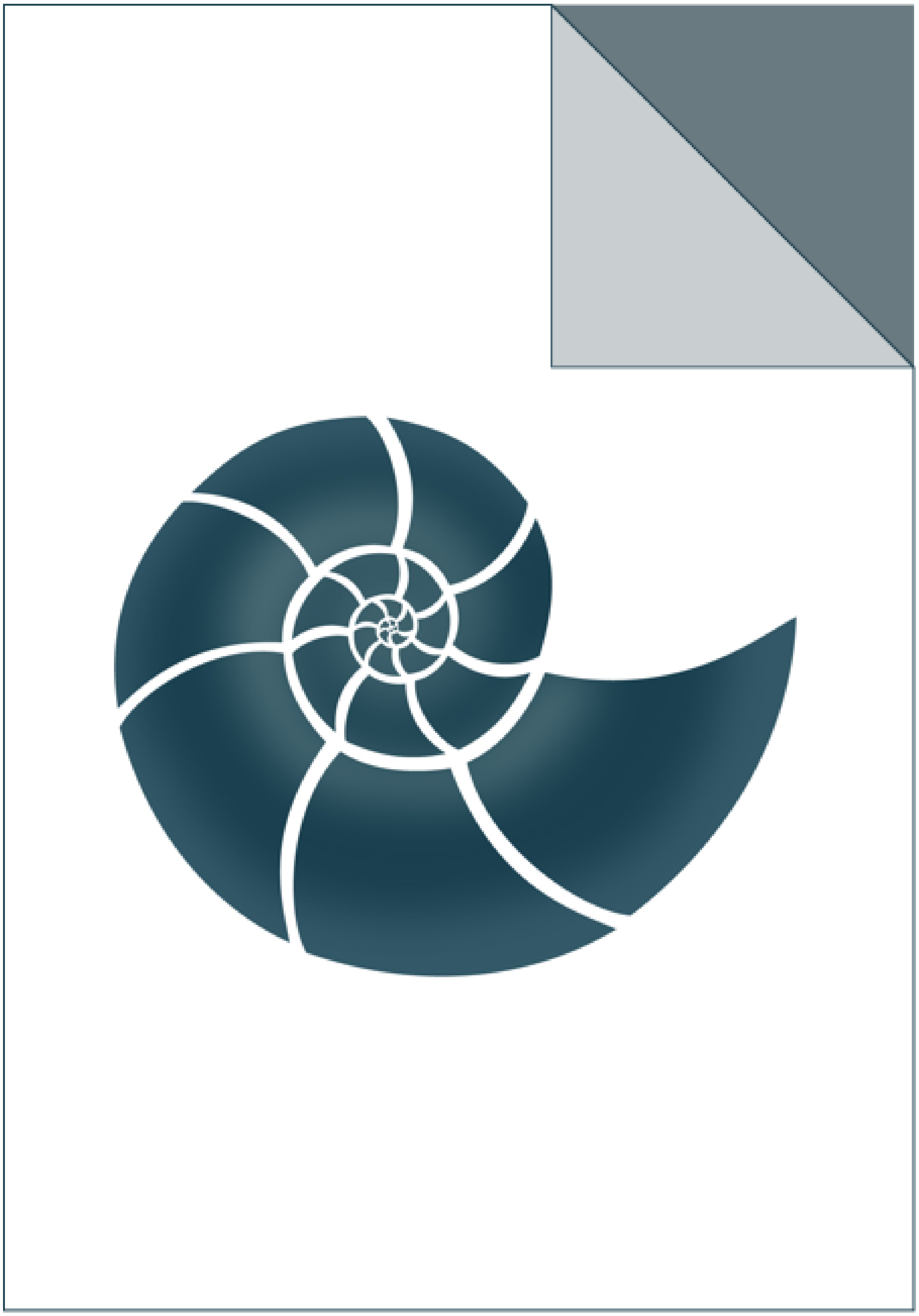
Demo for handling JSON data

Keywords:

- no_keywords

Categories:

- no_categories



5.2.31 ex_KDE_1D

Reads one column of observations and calculates Kernel Density Estimator (KDE) for the data

USAGE: ex_KDE_1D normal.txt 0.25 [min max periodic]

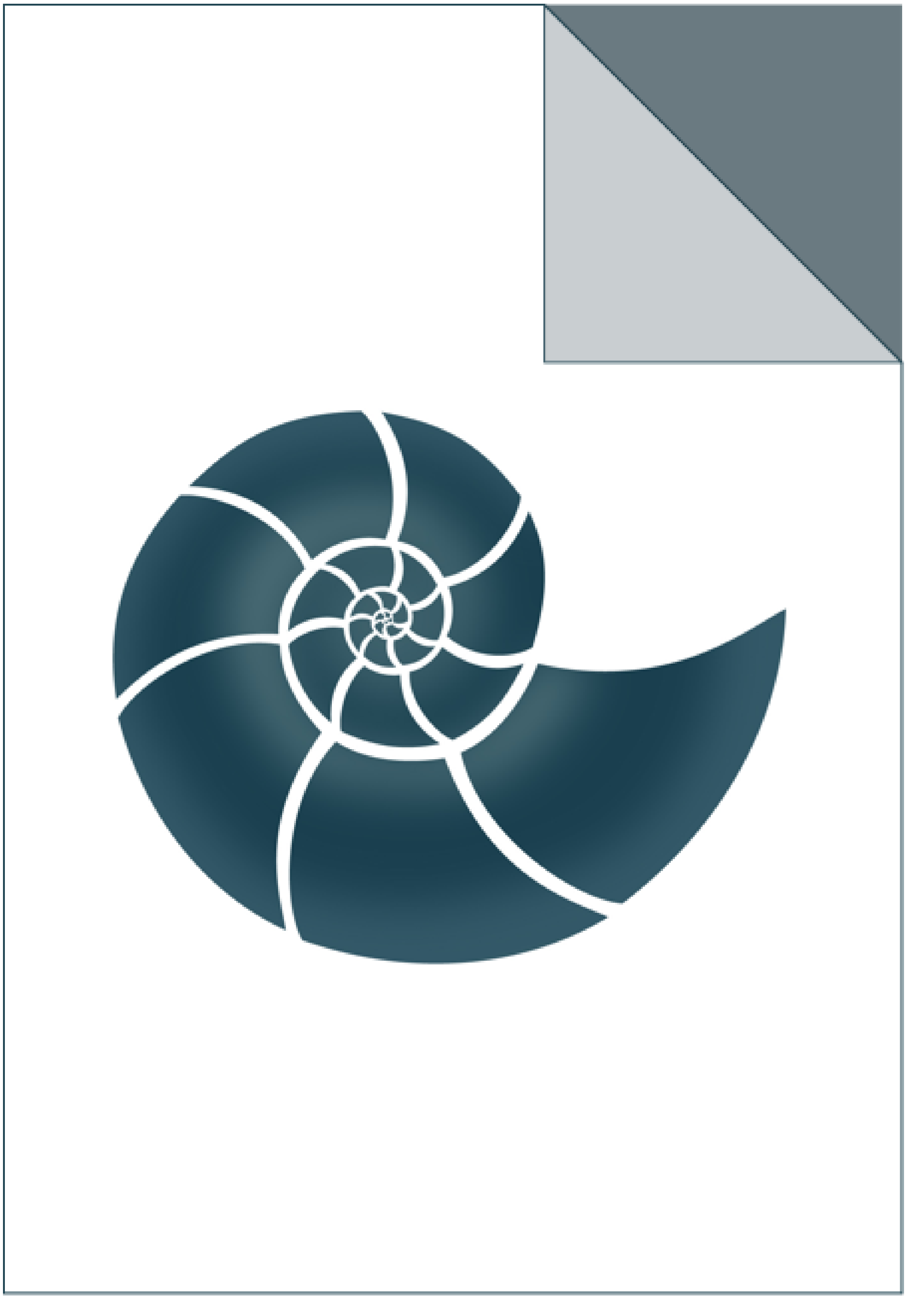
where normal.txt is the input file and 0.25 is the kernel bandwidth value. min and max are optional parameters to define the evaluation range. The last optional argument is the word 'periodic' to treat the estimated distribution as periodi

Keywords:

- no_keywords

Categories:

- no_categories



5.2.32 ex_LBFGS

Example shows how to use BFGS function minimizer

Keywords:

- no_keywords

Categories:

- no_categories

5.2.33 ex_Monomer

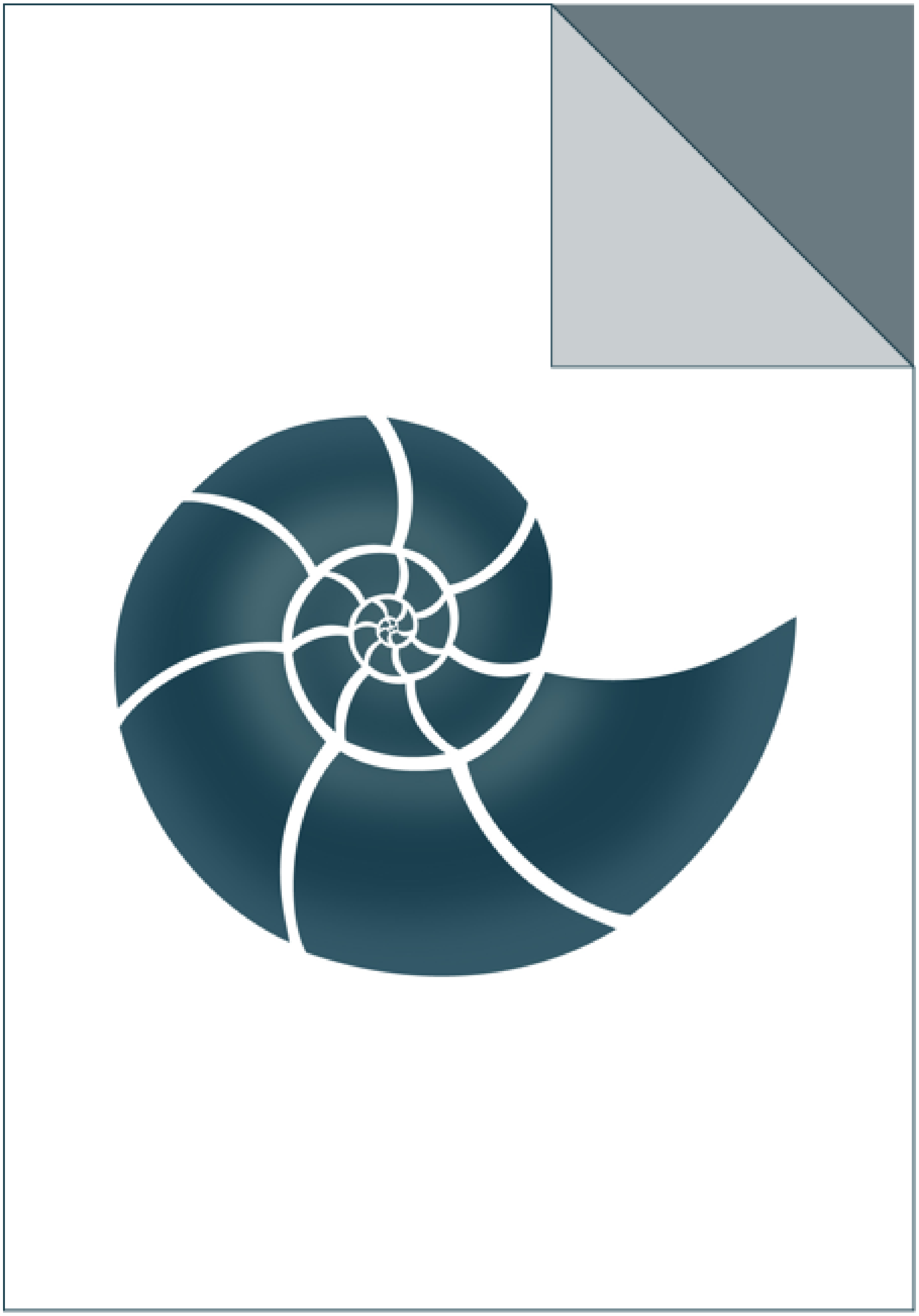
Example demonstrates functionality of core::chemical::Monomer data type.

Keywords:

- no_keywords

Categories:

- no_categories



5.2.34 ex_NormalDistribution

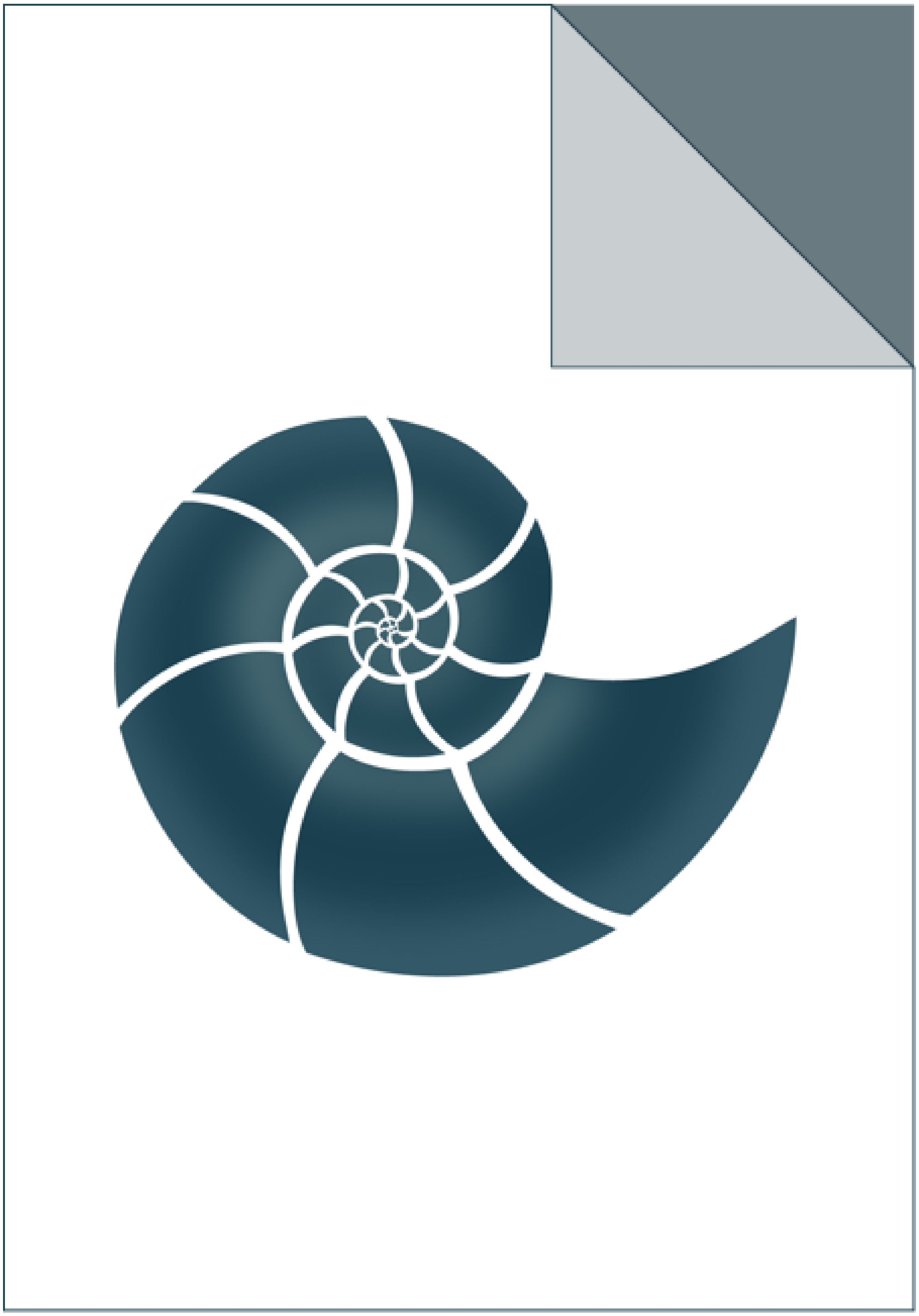
Demo for NormalDistribution class.

Keywords:

- no_keywords

Categories:

- no_categories



5.2.35 ex_OptionParser

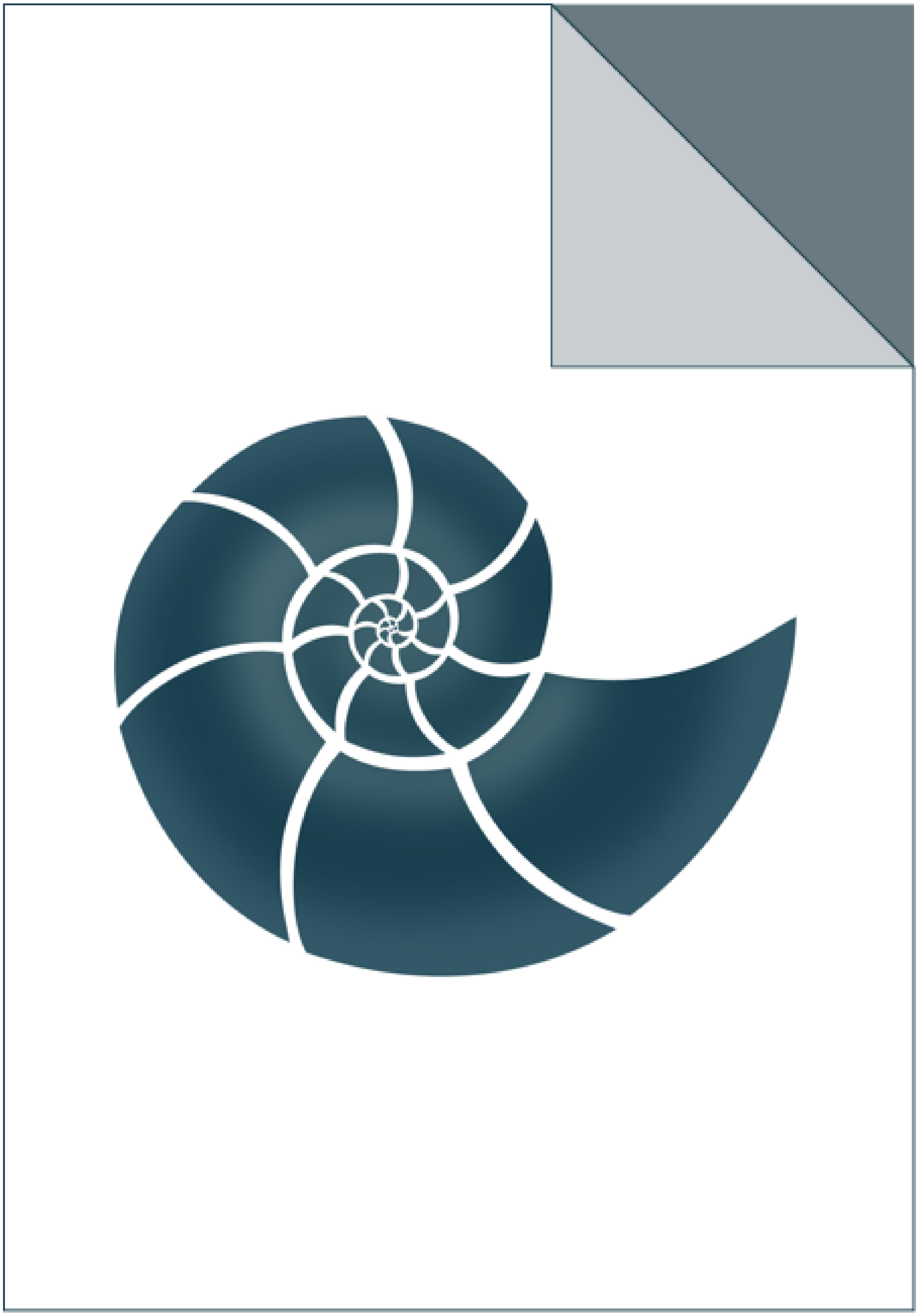
Shows how to use BioShell command line parser in your own program

Keywords:

- no_keywords

Categories:

- no_categories



5.2.36 ex_P2QuantileEstimation

ex_P2QuantileEstimation reads a file with real values and calculates a quantile using P-square algorithm

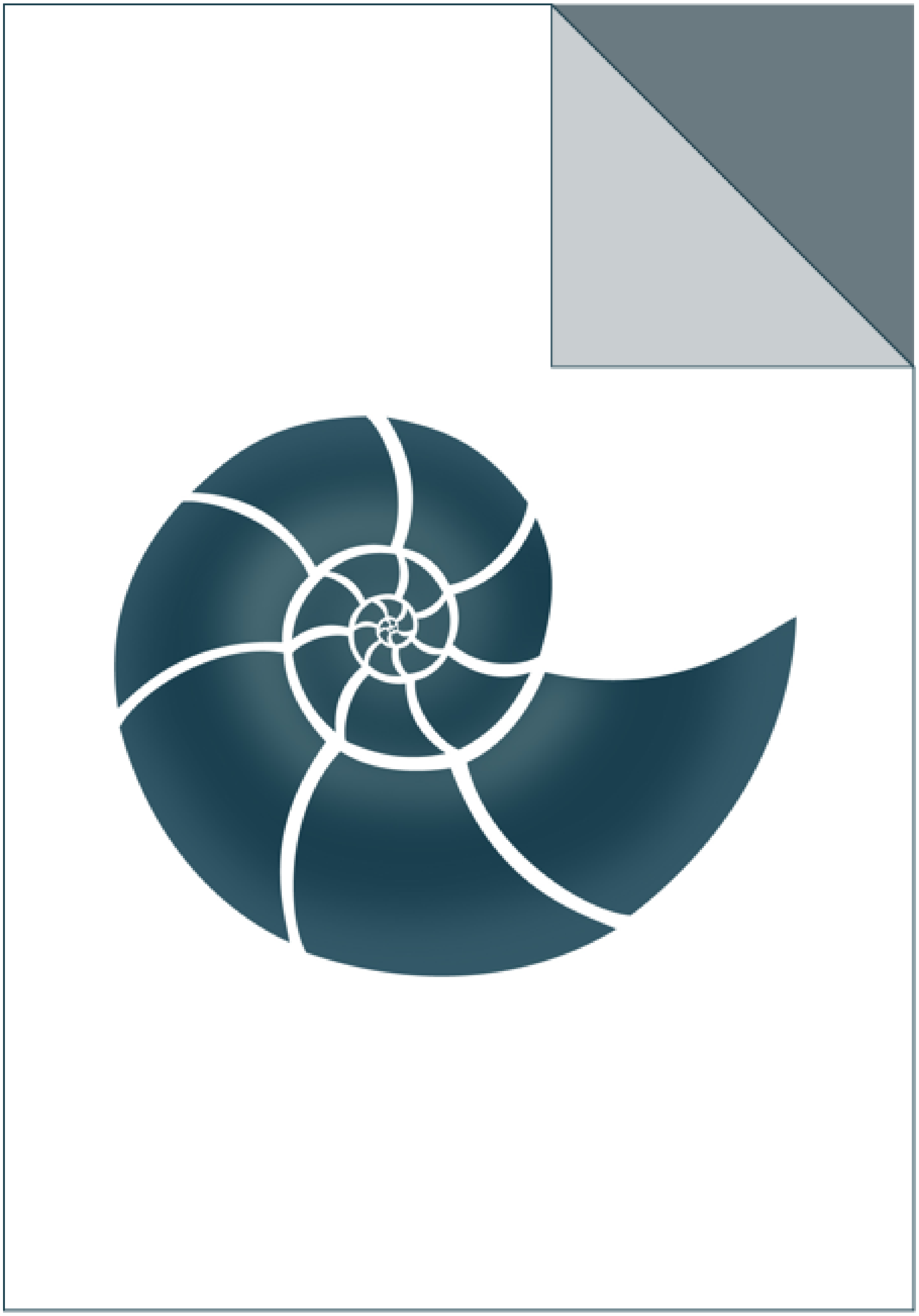
If no input file is provided, the program calculates 0.25, 0.5 and 0.75 quantiles of a random sample from normal distribution USAGE: ex_P2QuantileEstimation infile p_value

Keywords:

- no_keywords

Categories:

- no_categories



5.2.37 ex_PairwiseAlignment

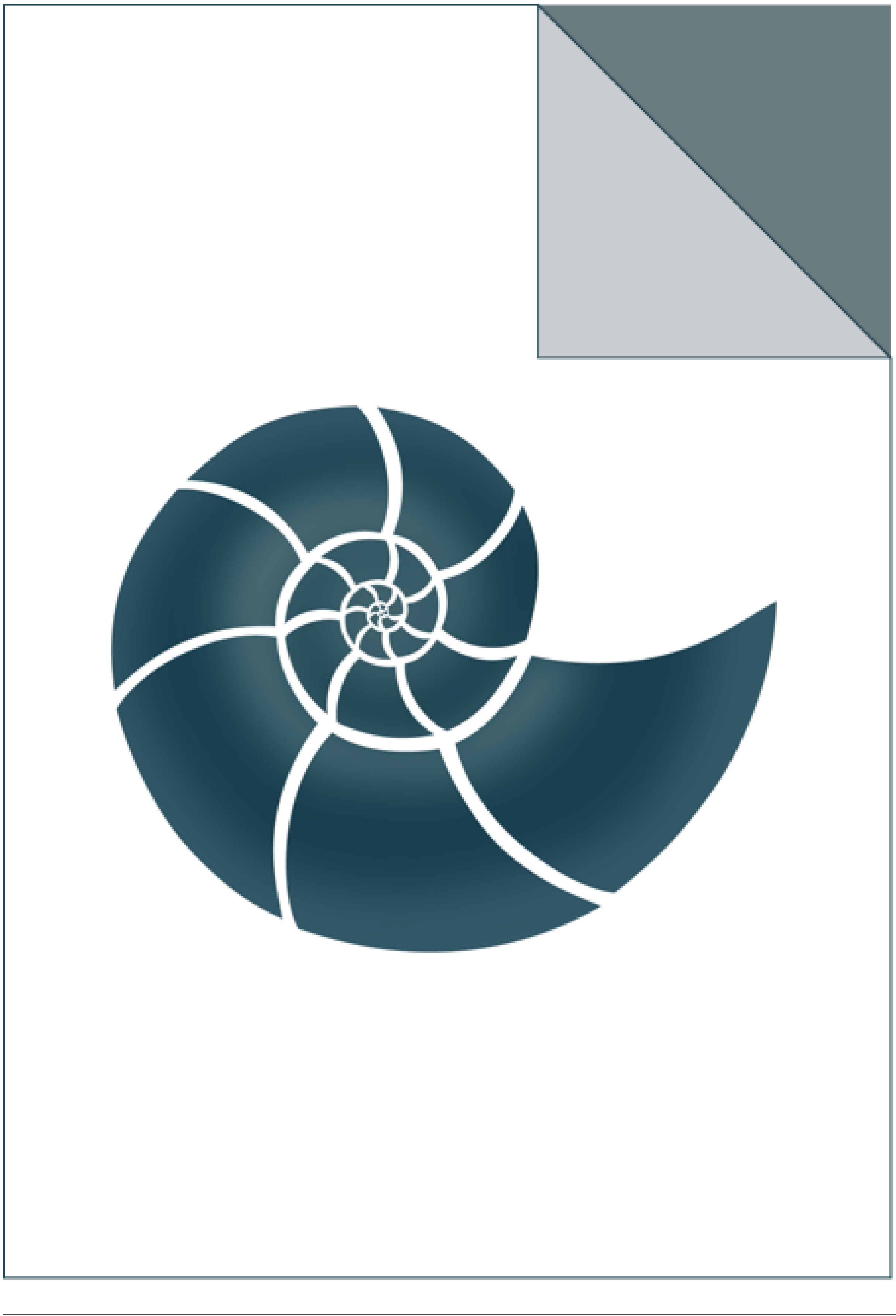
Simple example showing how to retrieve arbitrary data according to a sequence alignment object

Keywords:

- no_keywords

Categories:

- no_categories



5.2.38 ex_PairwiseSequenceAlignment

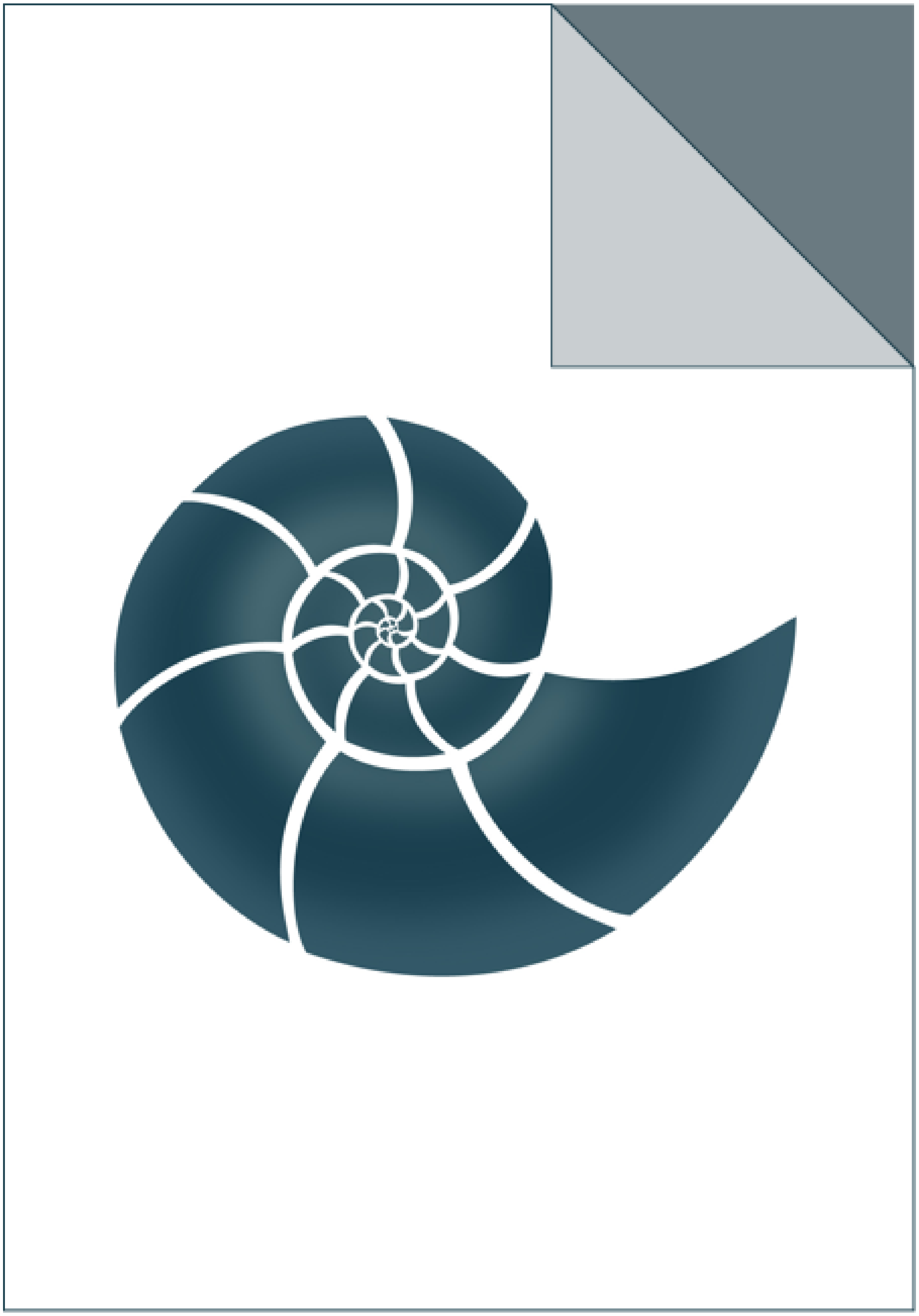
Calculates the optimal global sequence alignment between two protein sequences.

Keywords:

- no_keywords

Categories:

- no_categories



5.2.39 ex_Pca3

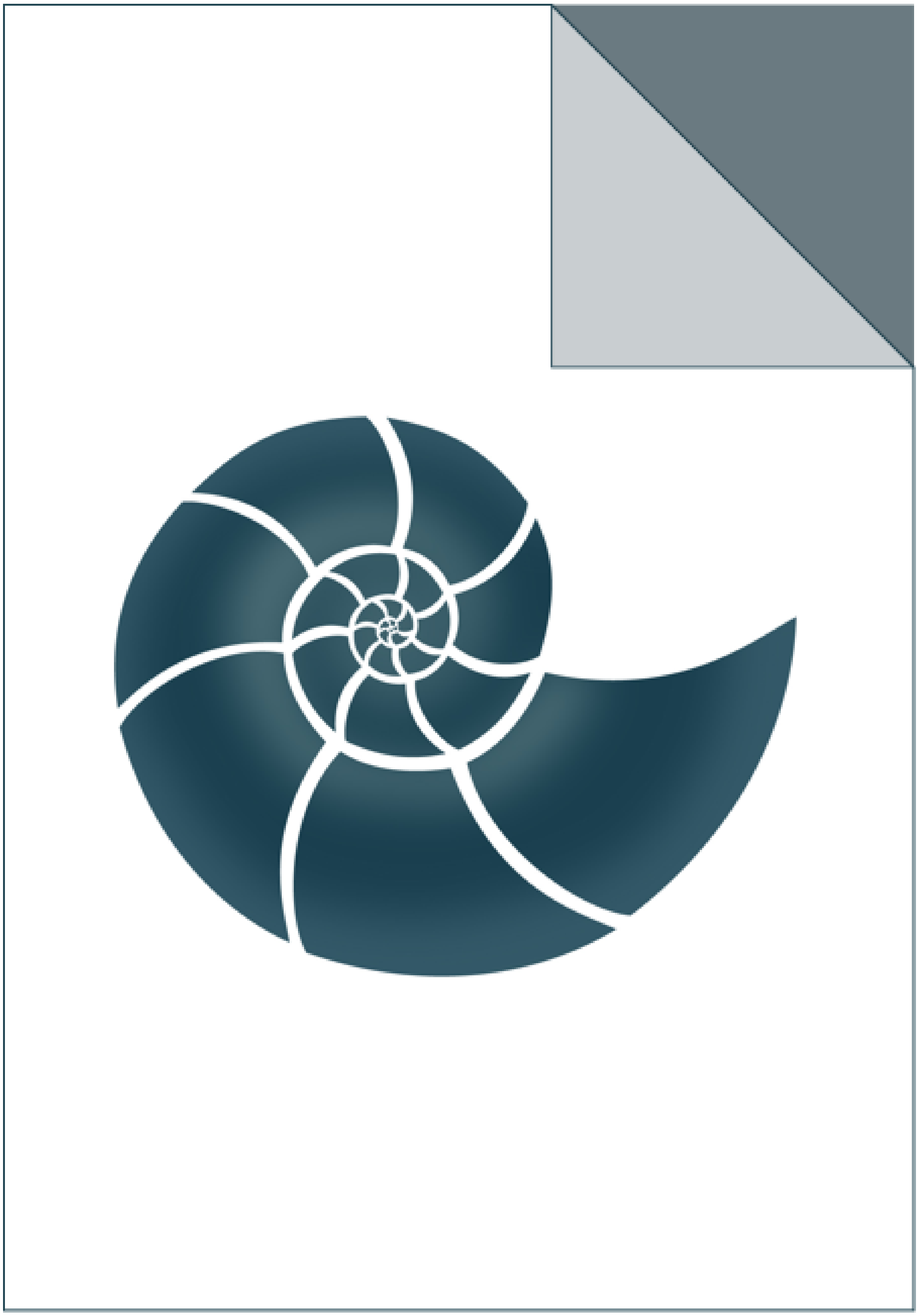
Orients 3D points along the axes using PCA algorithm

Keywords:

- no_keywords

Categories:

- no_categories



5.2.40 ex_Pdb

ex_Pdb demo shows how to read a PDB file and create a Structure object.

The program reads a given file with a PDB line filter that passes only backbone atoms and prints sone statistics about the input file

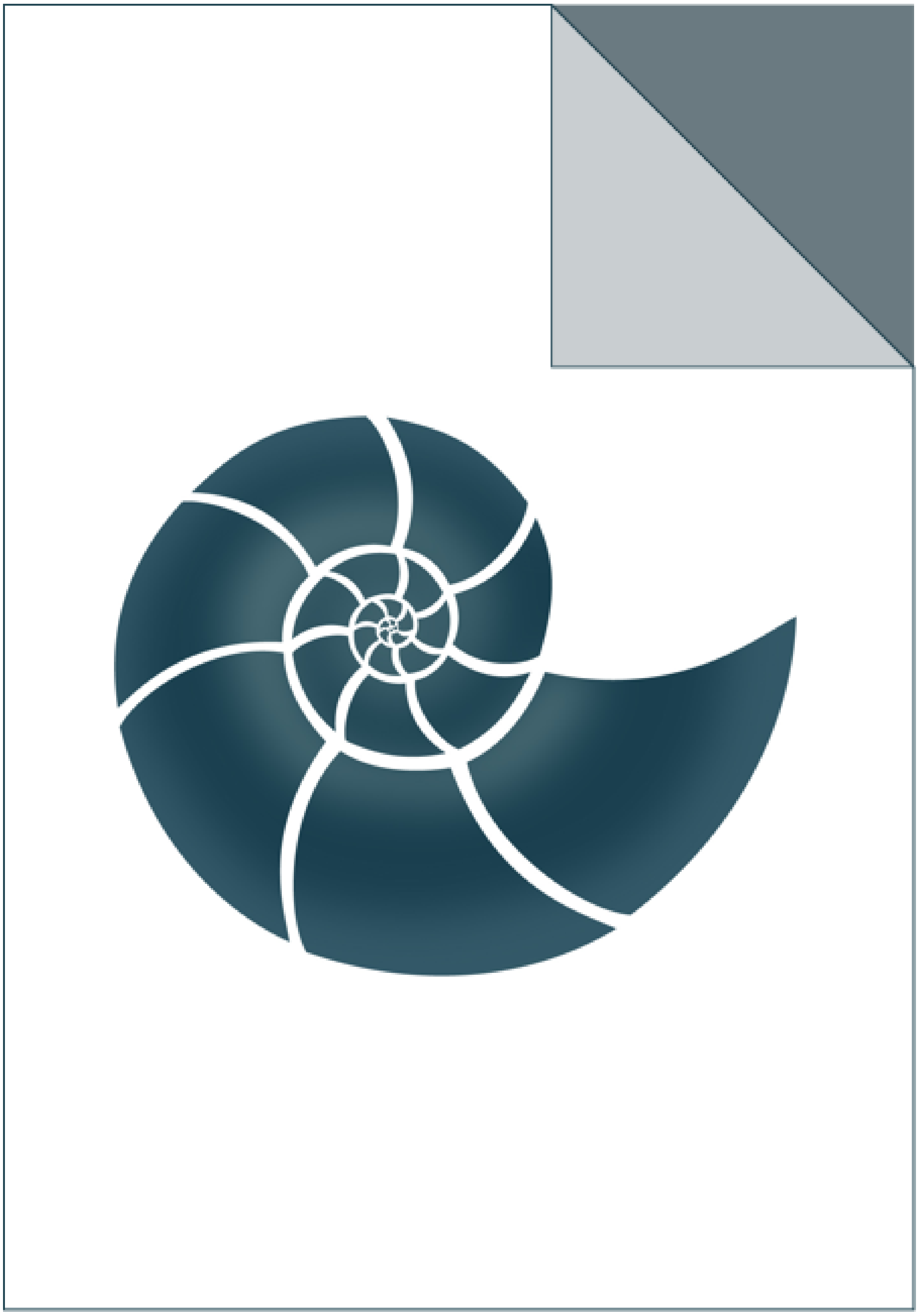
USAGE: ex_Pdb 5edw.pdb

Keywords:

- no_keywords

Categories:

- no_categories



5.2.41 ex_PdbLineFilter

Reads a PDB file and removes waters and alternate atom locations.

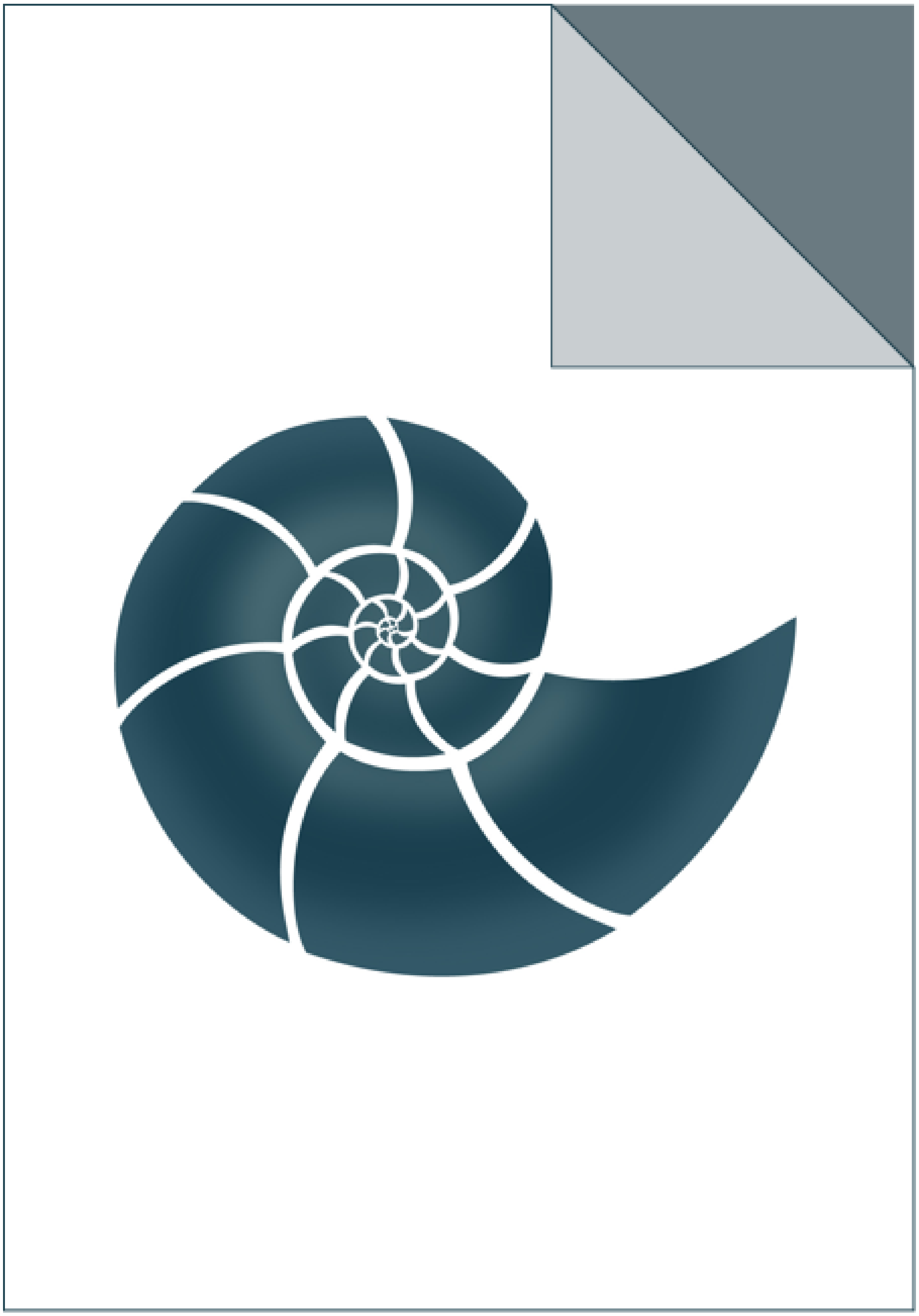
USAGE: ex_PdbLineFilter 5edw.pdb

Keywords:

- no_keywords

Categories:

- no_categories



5.2.42 ex_Quaternion

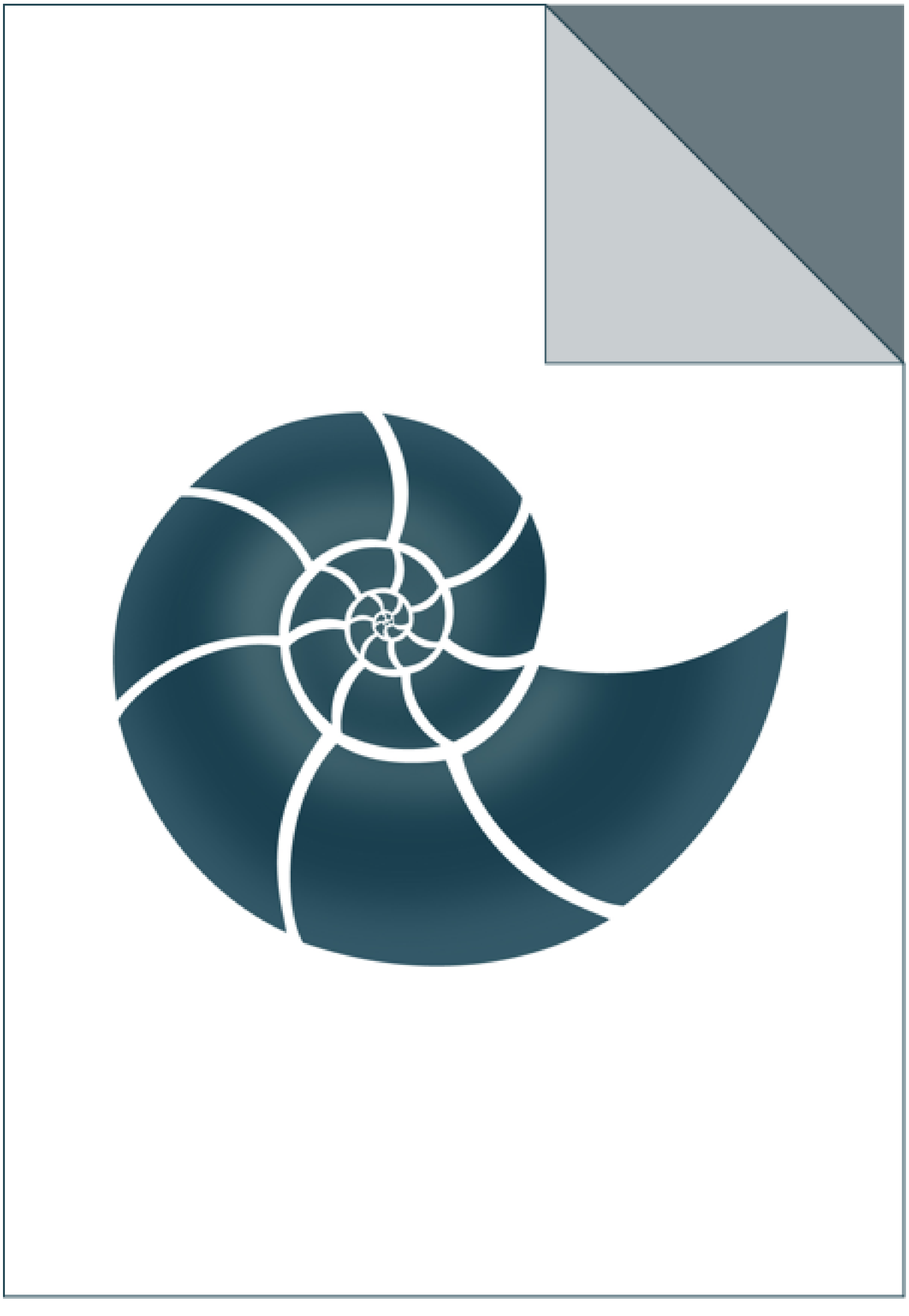
ex_Quaternion illustrates how to use Quaternion class

Keywords:

- no_keywords

Categories:

- no_categories



5.2.43 ex_ReduceSequenceAlphabet

> If no input is given, ex_ReduceSequenceAlphabet lists all reduced amino acid alphabets registered in BioShell library.

Alternatively, user can provide an alphabet name; in this cace the relevant mapping is printed on the screen.

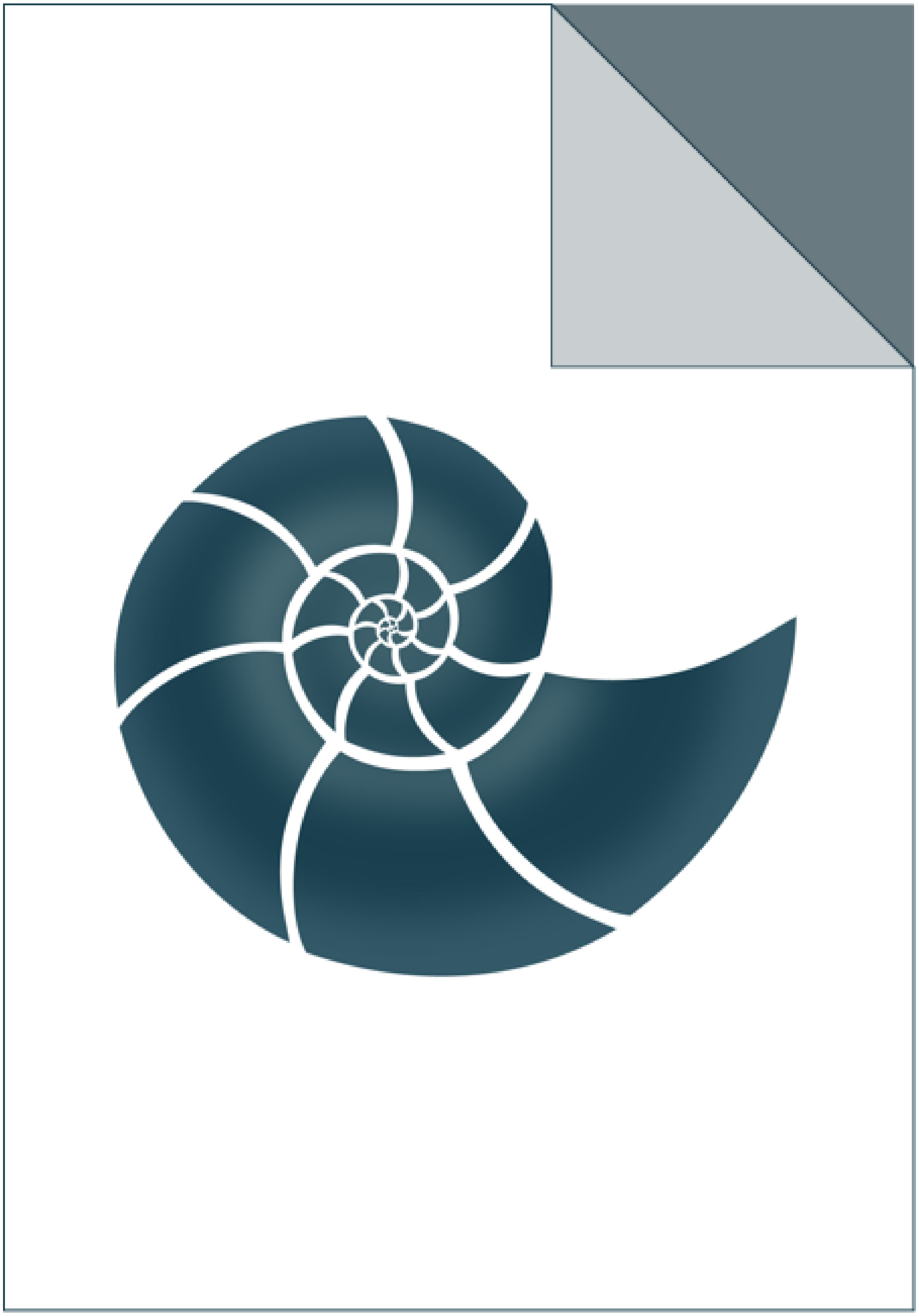
USAGE: ex_ReduceSequenceAlphabet [alphabet_name]

Keywords:

- no_keywords

Categories:

- no_categories



5.2.44 ex_Remark290

ex_Pdb demo shows how to access symmetry operators stored in a PDB file header.

The program reads a given PDB file and prints all cymmetry operators as rototranslation objects

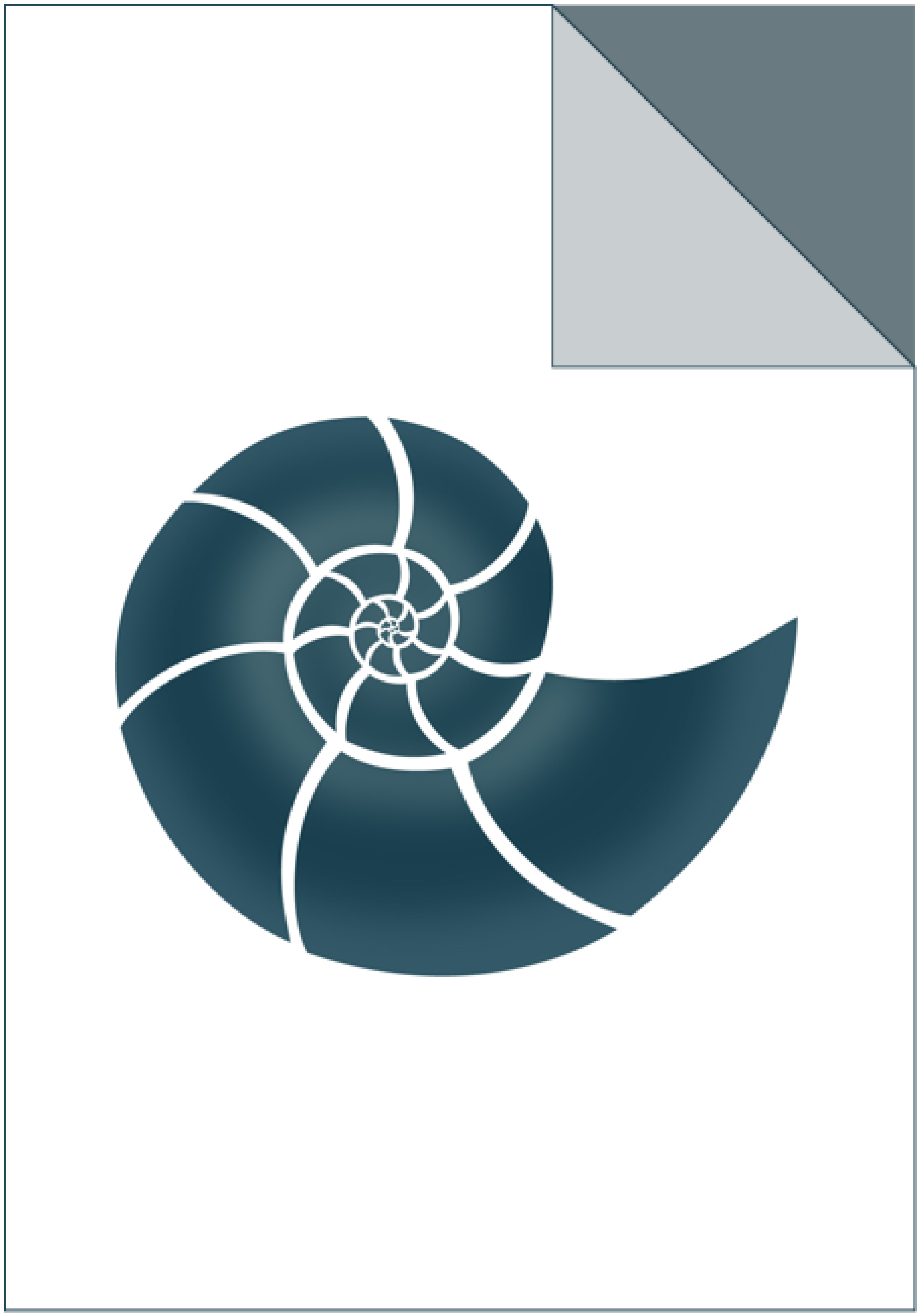
USAGE: ex_Remark290 5edw.pdb

Keywords:

- no_keywords

Categories:

- no_categories



5.2.45 ex_Residue

ex_Residue reads a PDB file and checks if all amino acid residues have complete backbone

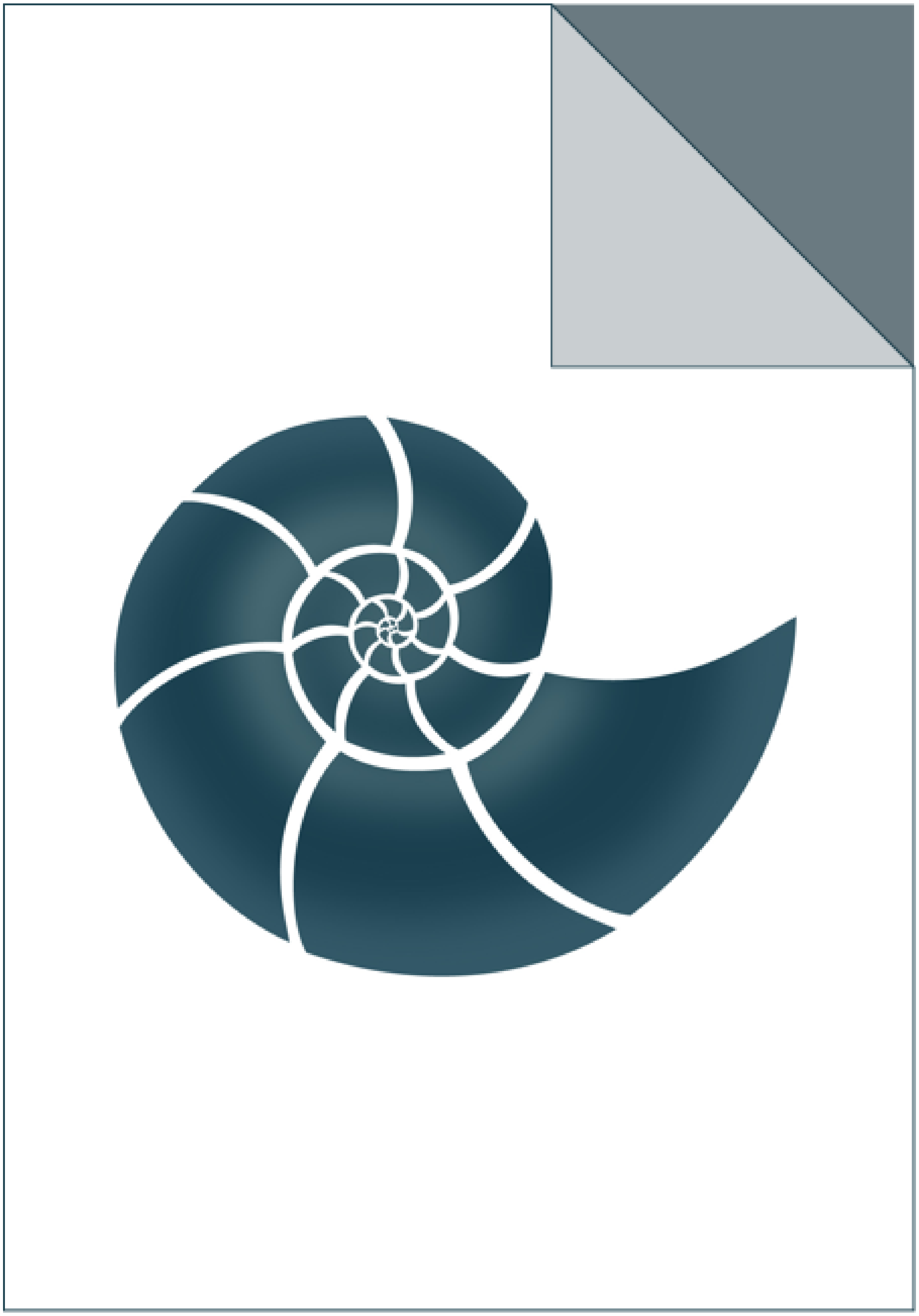
USAGE: ex_Residue 5edw.pdb

Keywords:

- no_keywords

Categories:

- no_categories



5.2.46 ex_RobustDistributionDecorator

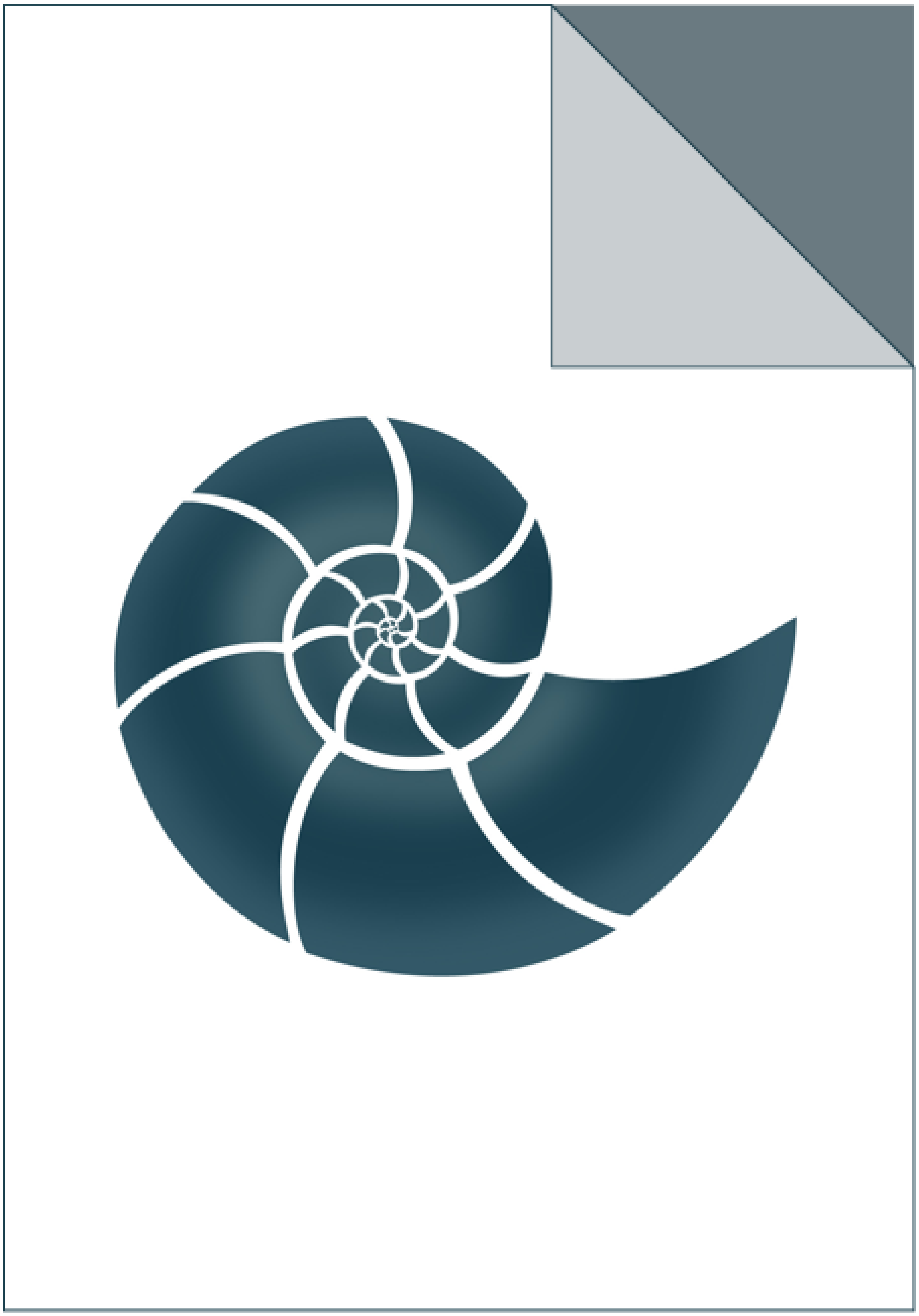
Example showing how to create and use a RobustDistributionDecorator

Keywords:

- no_keywords

Categories:

- no_categories



5.2.47 ex_SelectChainBreaks

Reads a PDB file and prints list of chain breaks found in every chain

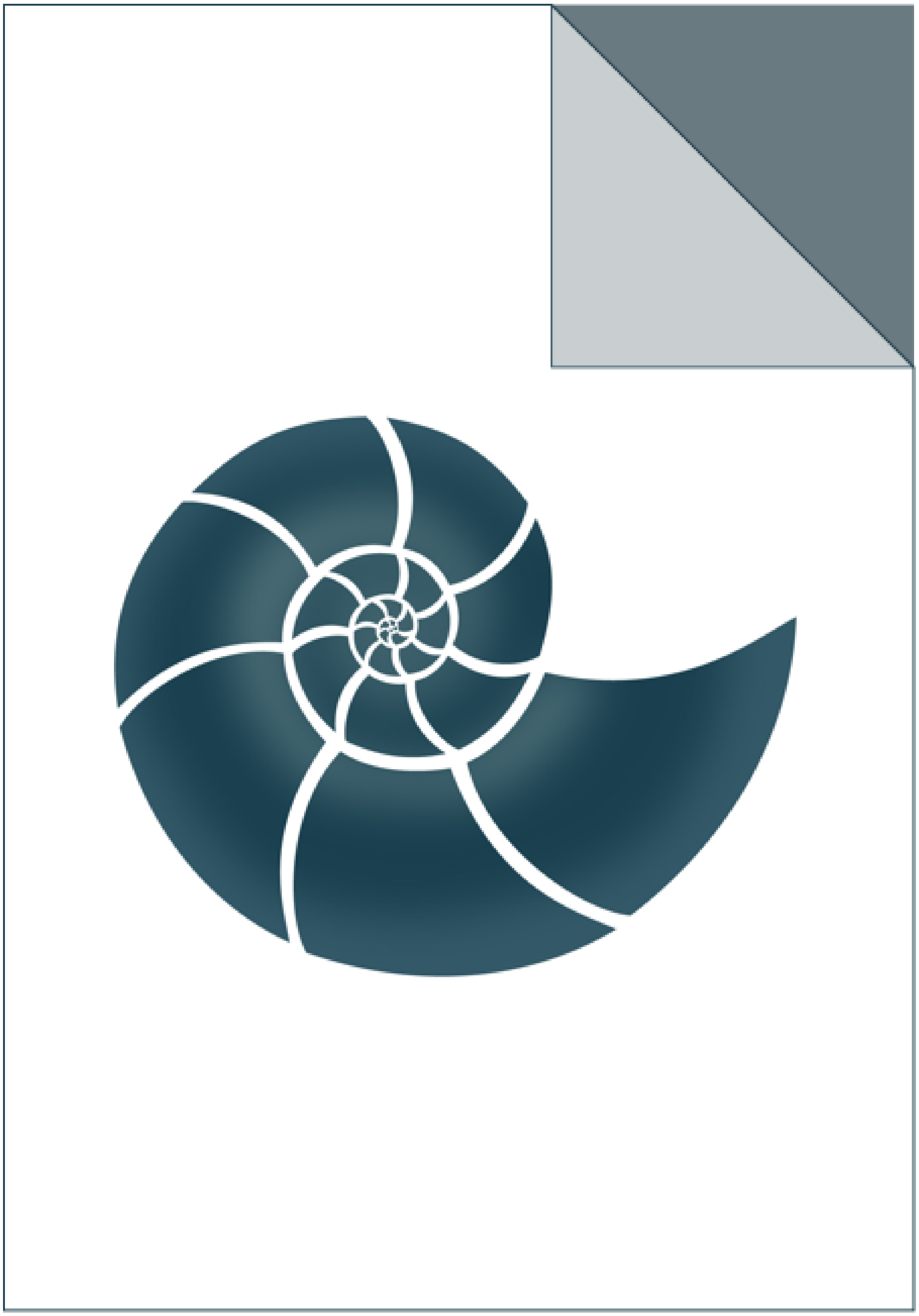
USAGE: ex_SelectChainBreaks test_inputs/4mcb.pdb

Keywords:

- no_keywords

Categories:

- no_categories



5.2.48 ex_SelectResidueRange

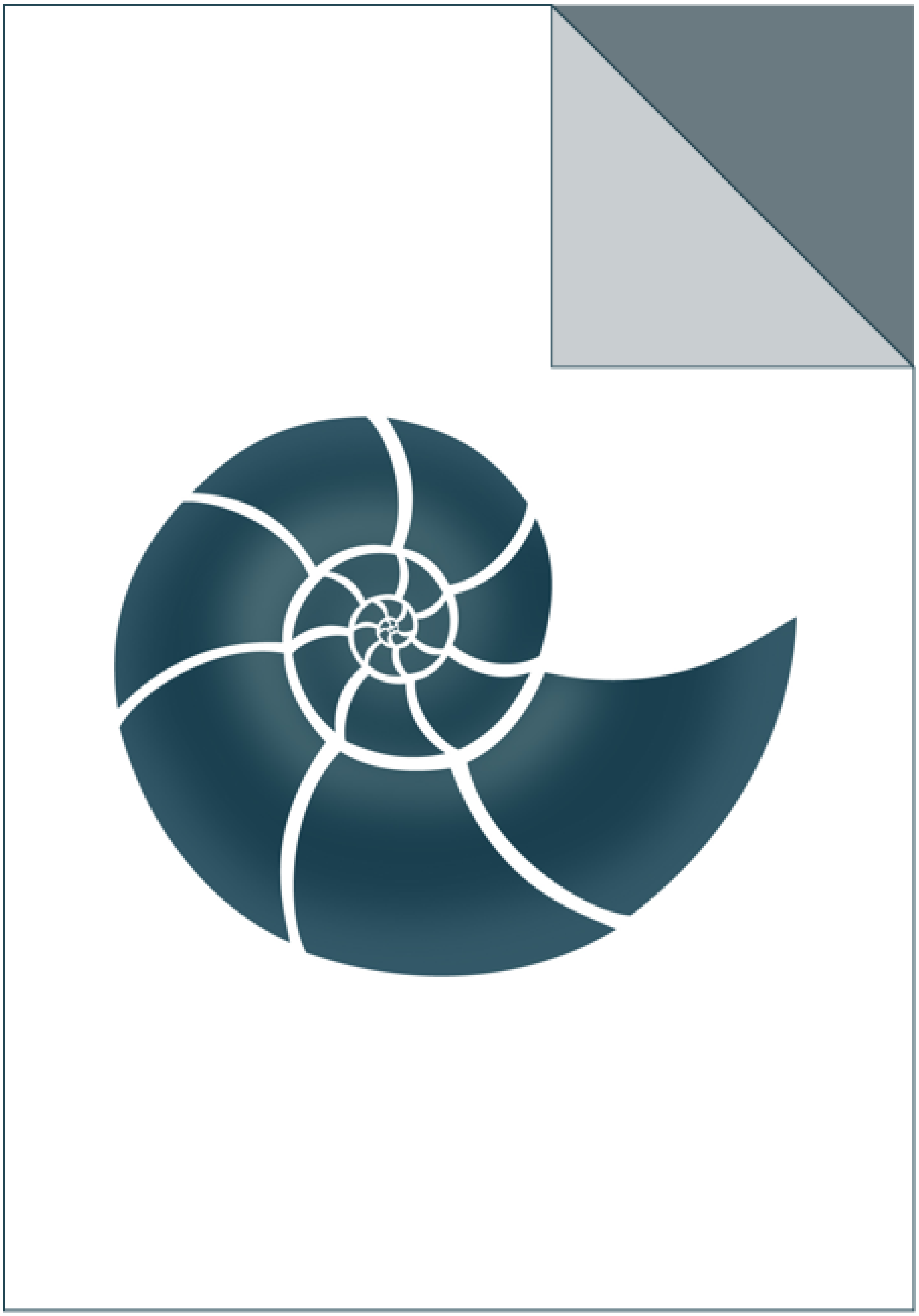
Shows how to select a structural fragment based on residue IDs

Keywords:

- no_keywords

Categories:

- no_categories



5.2.49 ex_SemiglobalAligner

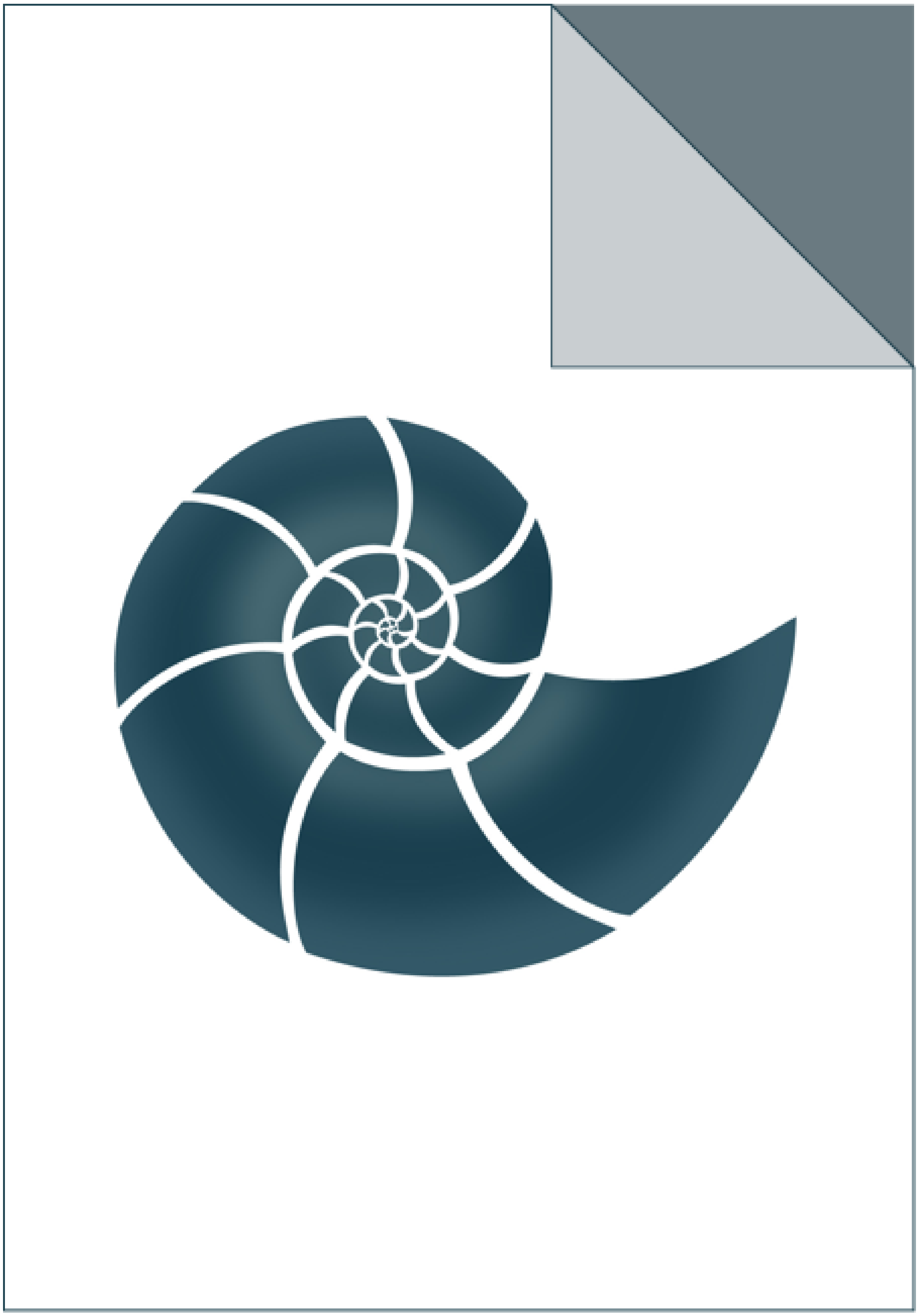
Calculate a pairwise sequence alignment between two sequences with identity scoring method.

Keywords:

- no_keywords

Categories:

- no_categories



5.2.50 ex_Seqres

ex_Seqres reads a PDB file and prints the sequences stored in its SEQRES fields

These sequences in many cases differ ftom the sequences extracted from coordinates section

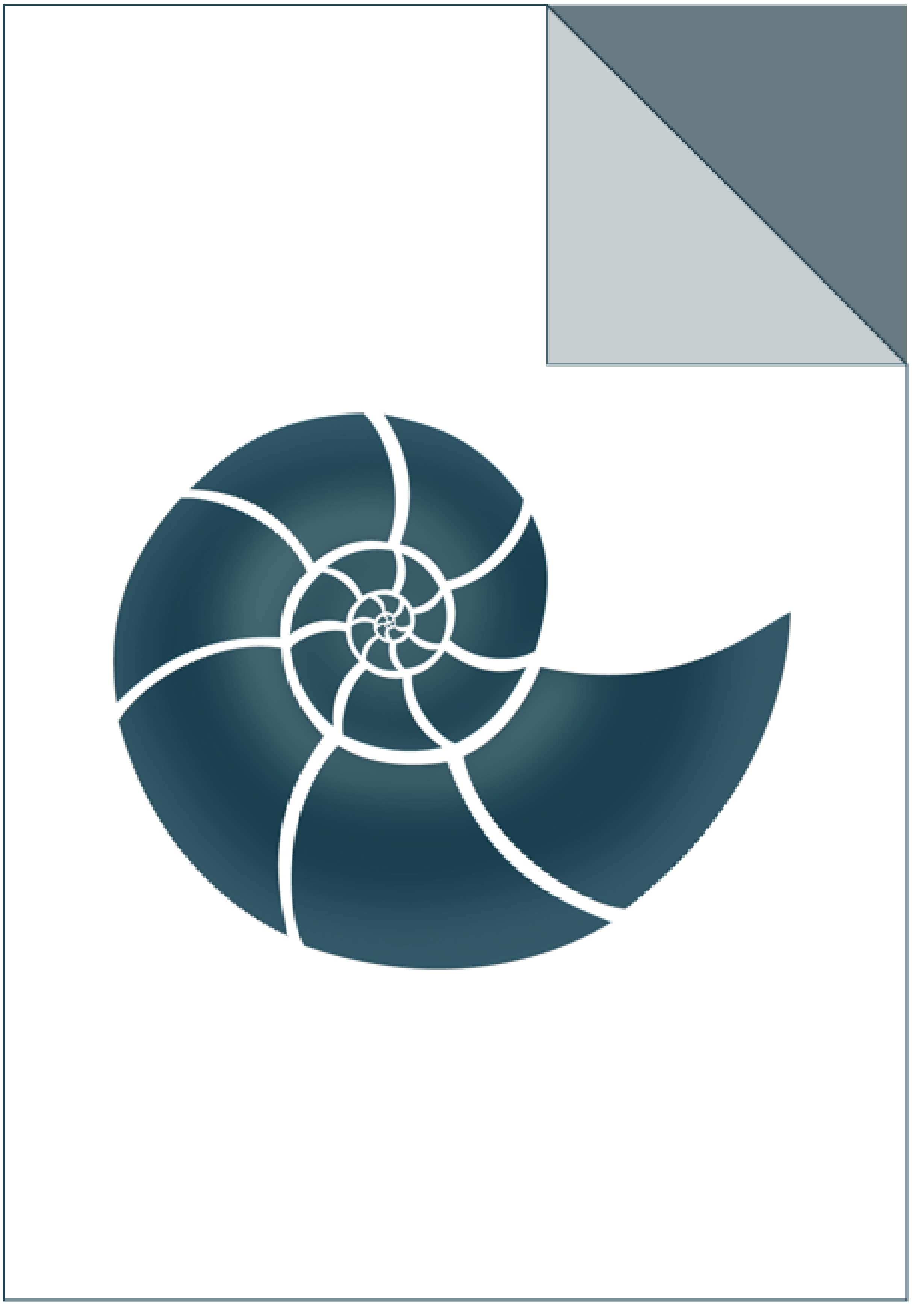
USAGE: ./ex_Seqres 2kwi.pdb

Keywords:

- no_keywords

Categories:

- no_categories



5.2.51 ex_Sequence

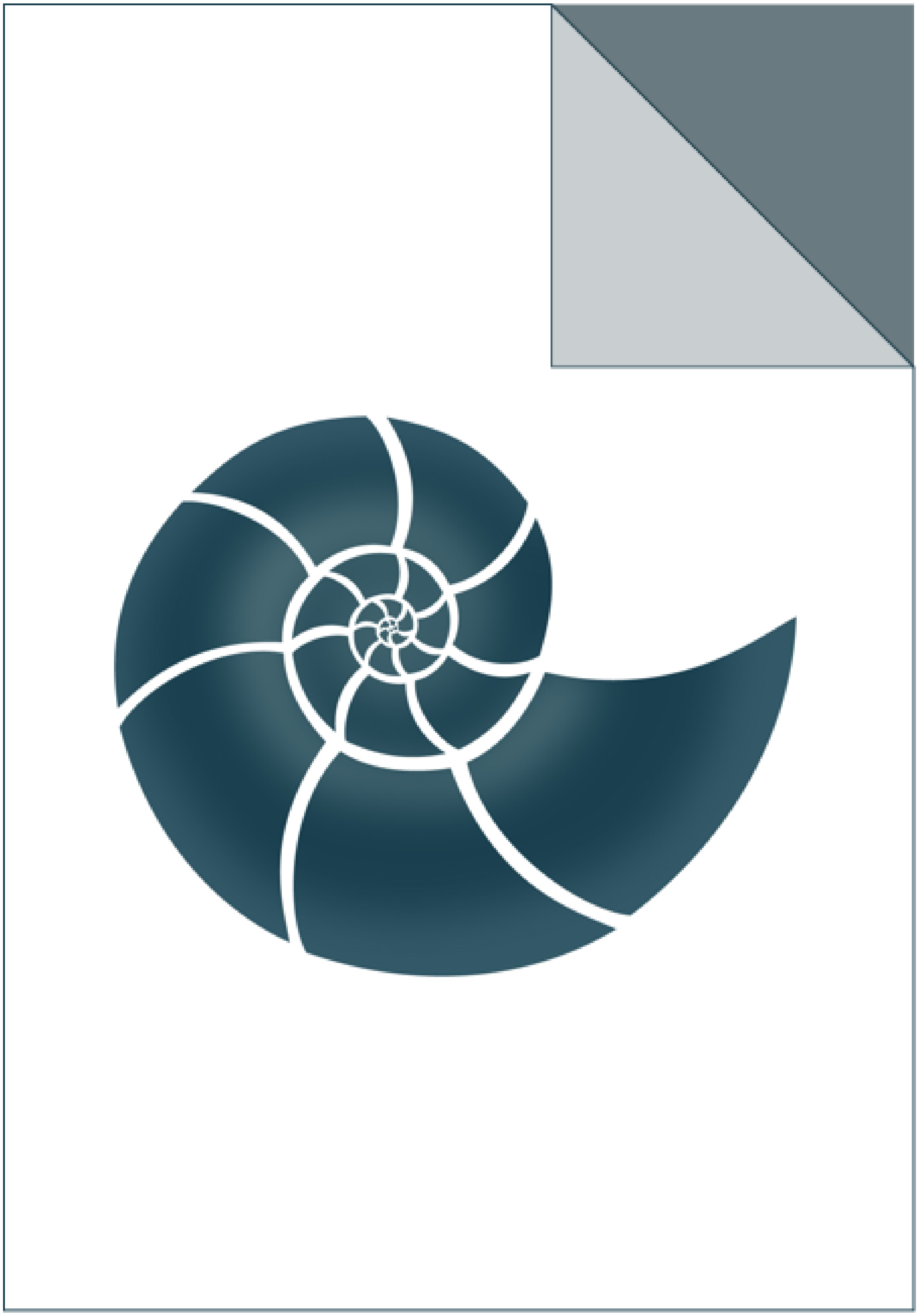
ex_Sequence reads a PDB file and prints a sequence fragment. It demonstrates
how to select residues by their PDB_ID USAGE: ./ex_Sequence 3wn7.pdb

Keywords:

- no_keywords

Categories:

- no_categories



5.2.52 ex_SequenceAlignmentWidget

Simple test for SequenceAlignmentComponent web-component reads an alignment from a PIR or FASTA file

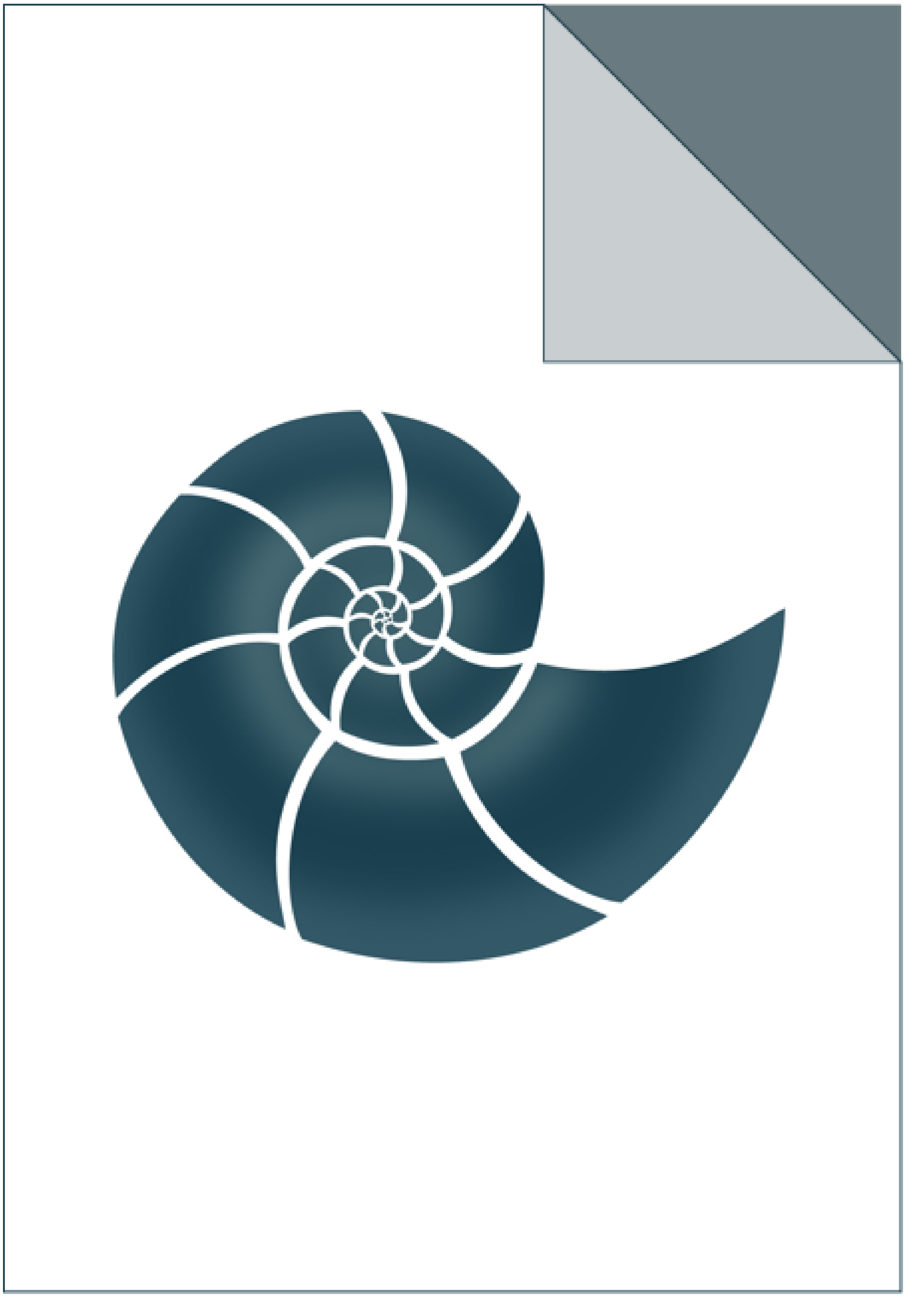
and formats it to HTM

Keywords:

- no_keywords

Categories:

- no_categories



5.2.53 ex_Structure

ex_Structure reads a PDB file and prints a list of all atoms grouped by residues they belong to

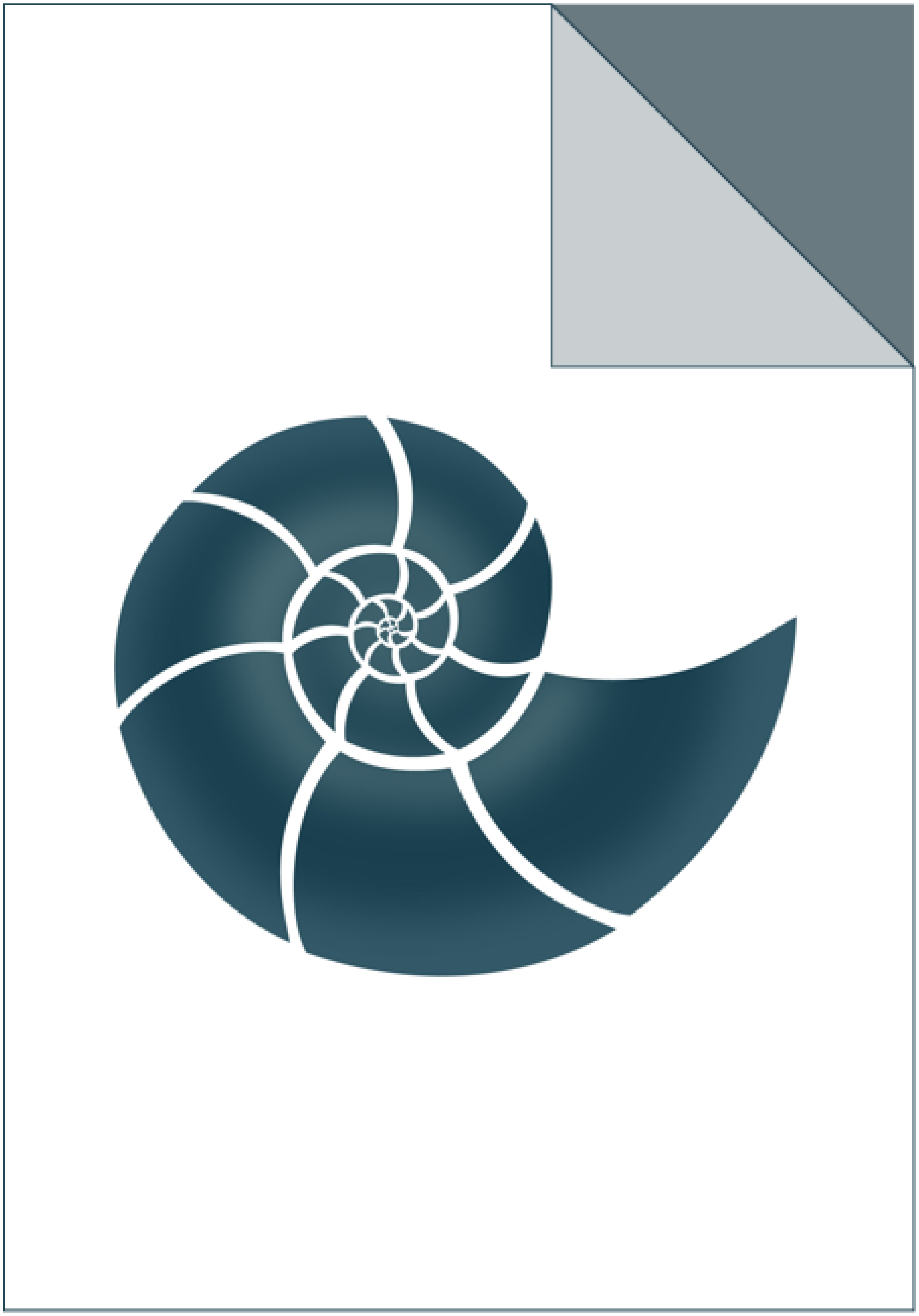
USAGE: ./ex_Structure 5edw.pdb

Keywords:

- no_keywords

Categories:

- no_categories



5.2.54 ex_ThreadPool

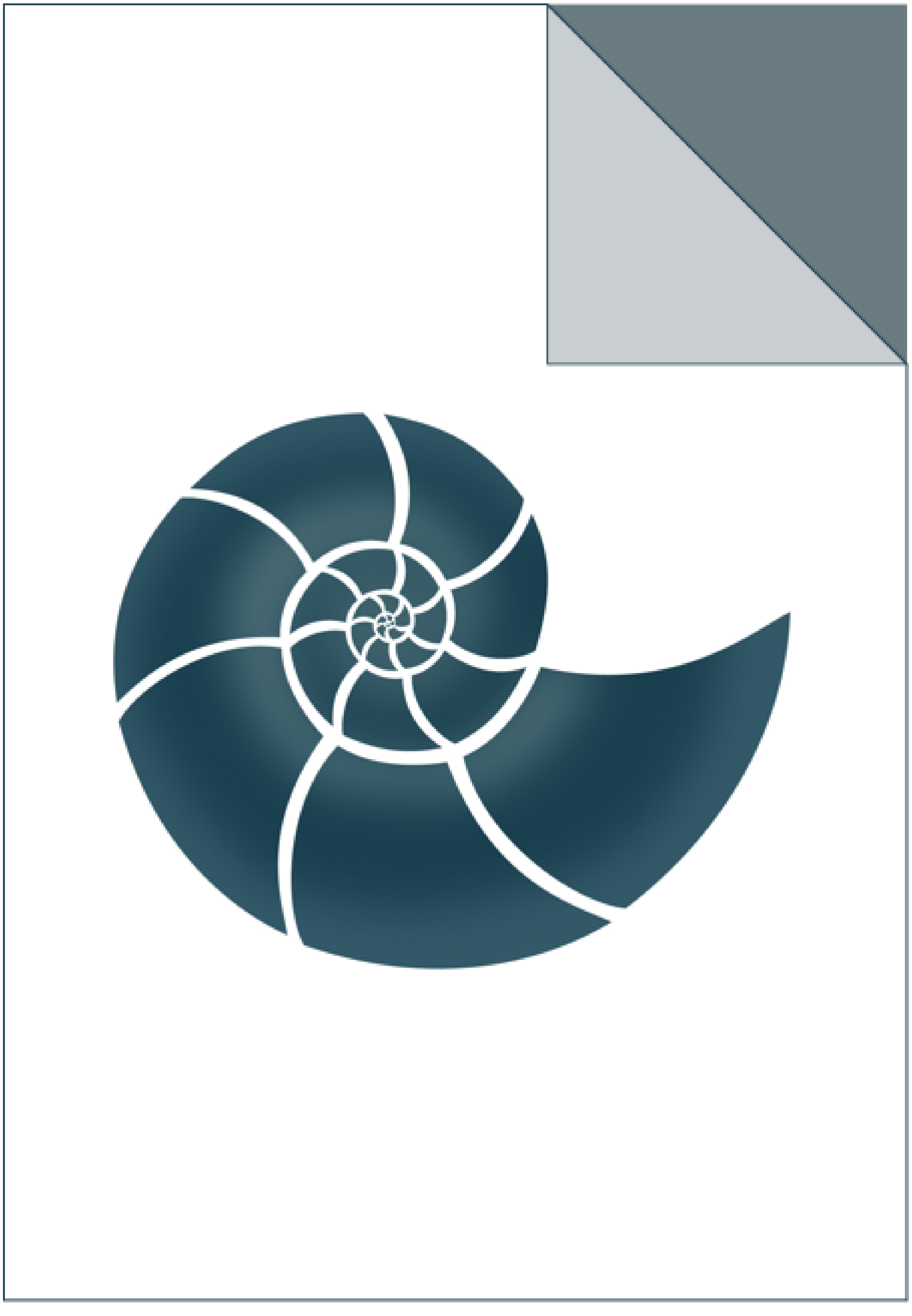
Simple test for a ThreadPool class

Keywords:

- no_keywords

Categories:

- no_categories



5.2.55 ex_ThreadSafeMap

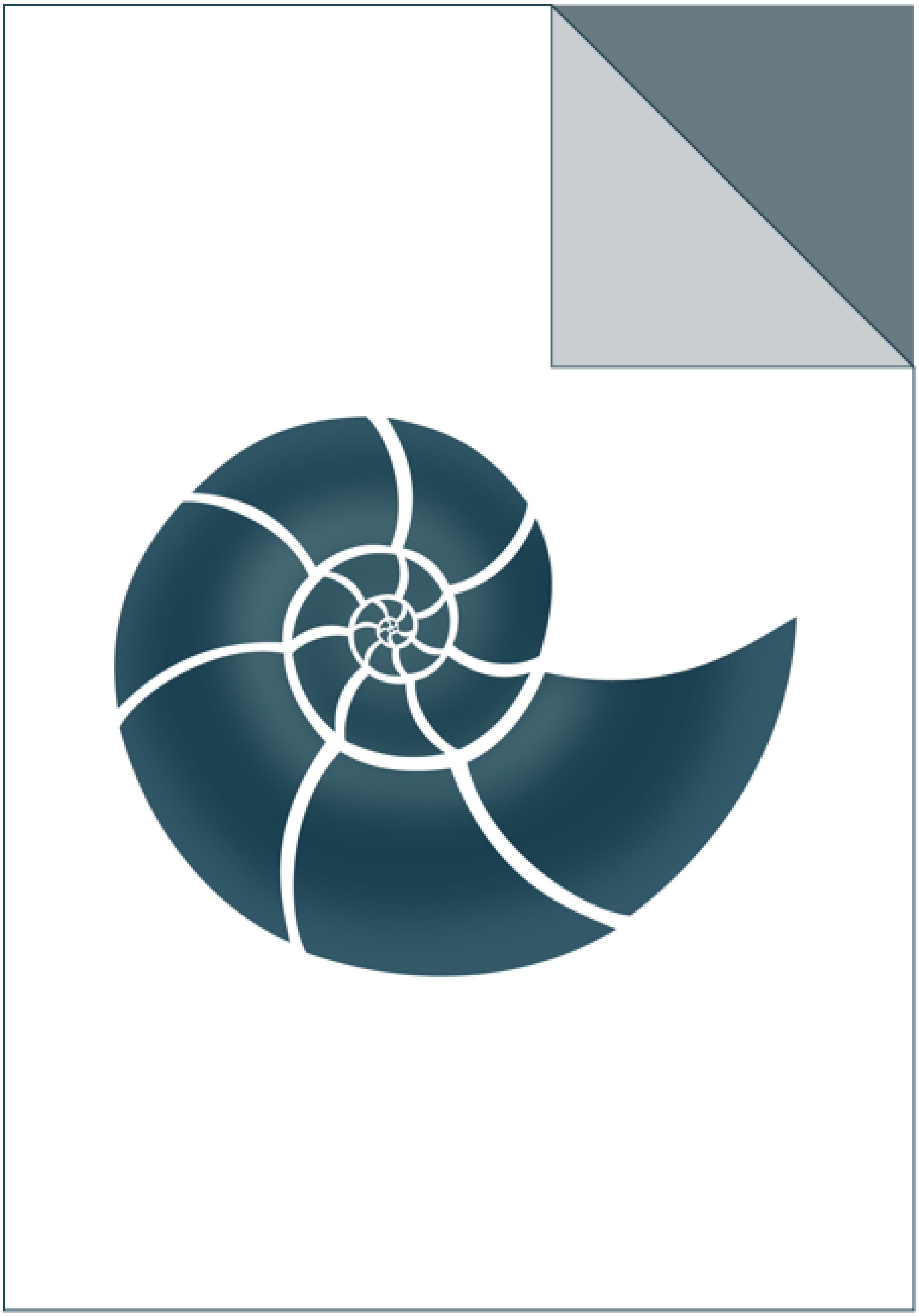
Shows how to use ThreadSafeMap class

Keywords:

- no_keywords

Categories:

- no_categories



5.2.56 ex_ThreeDTree

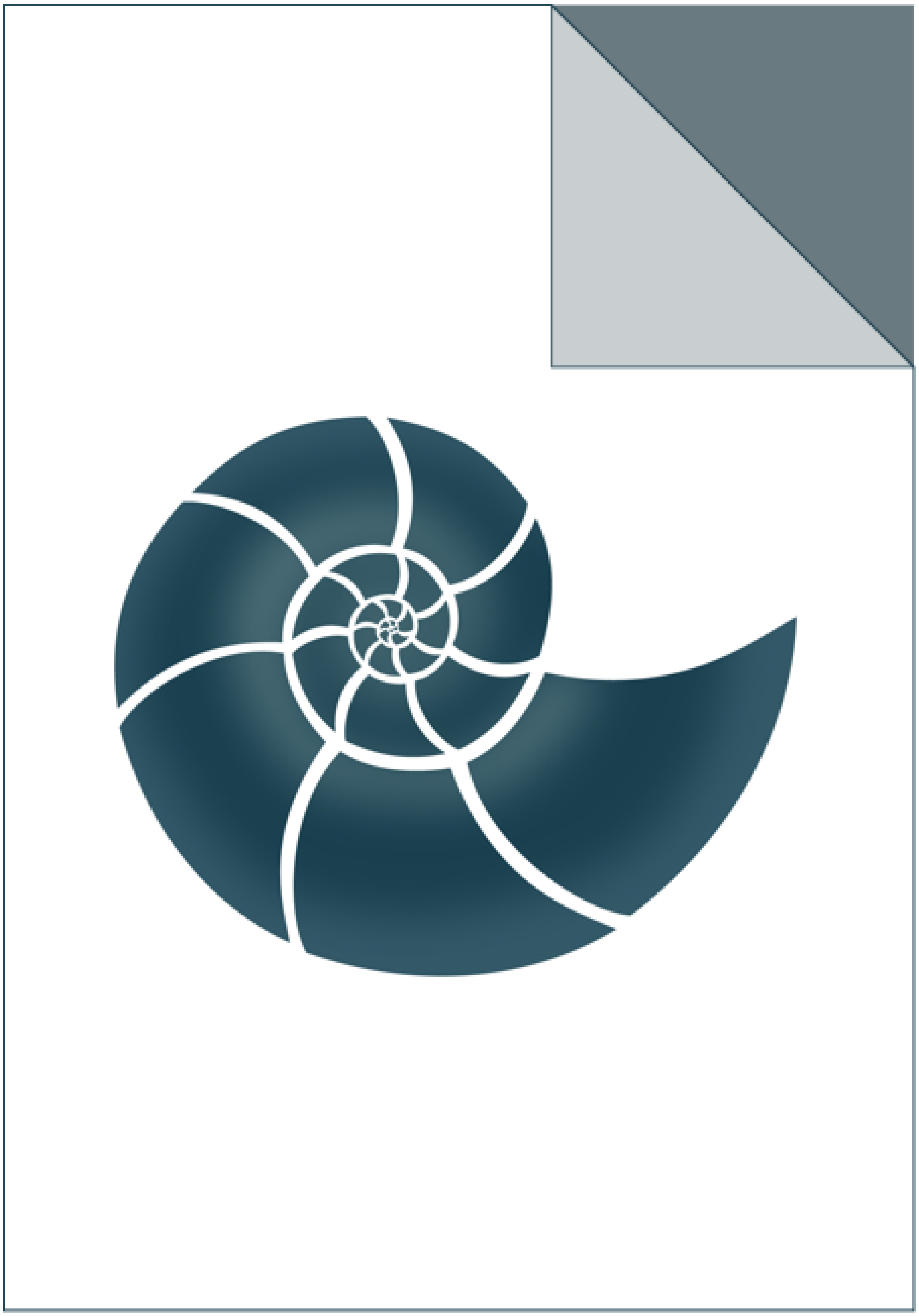
A simple example shows how to use BioShell kd-tree routines.

Keywords:

- no_keywords

Categories:

- no_categories



5.2.57 ex_TreeNode

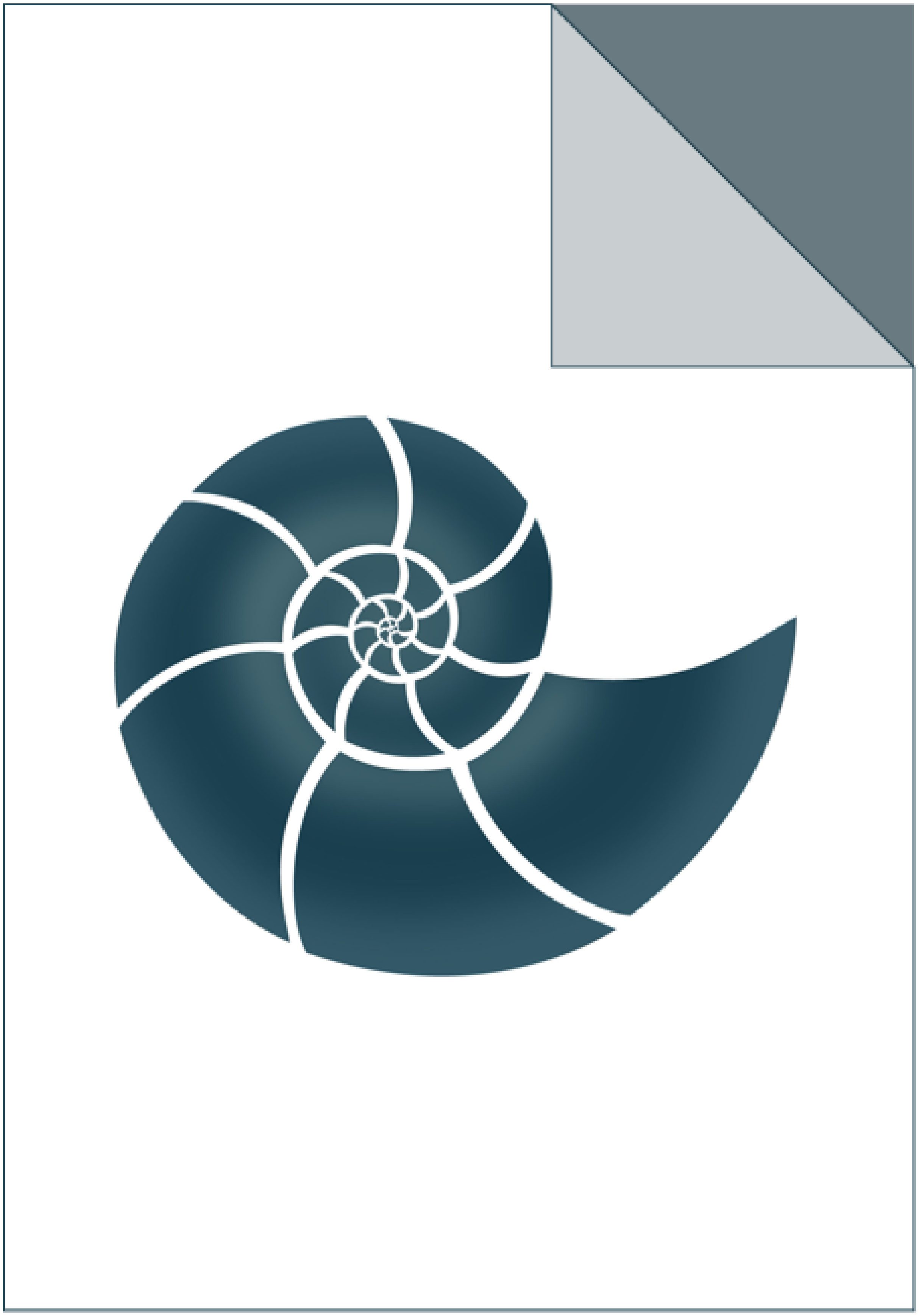
Simple demo for TreeNode class

Keywords:

- no_keywords

Categories:

- no_categories



5.2.58 ex_UnionFind

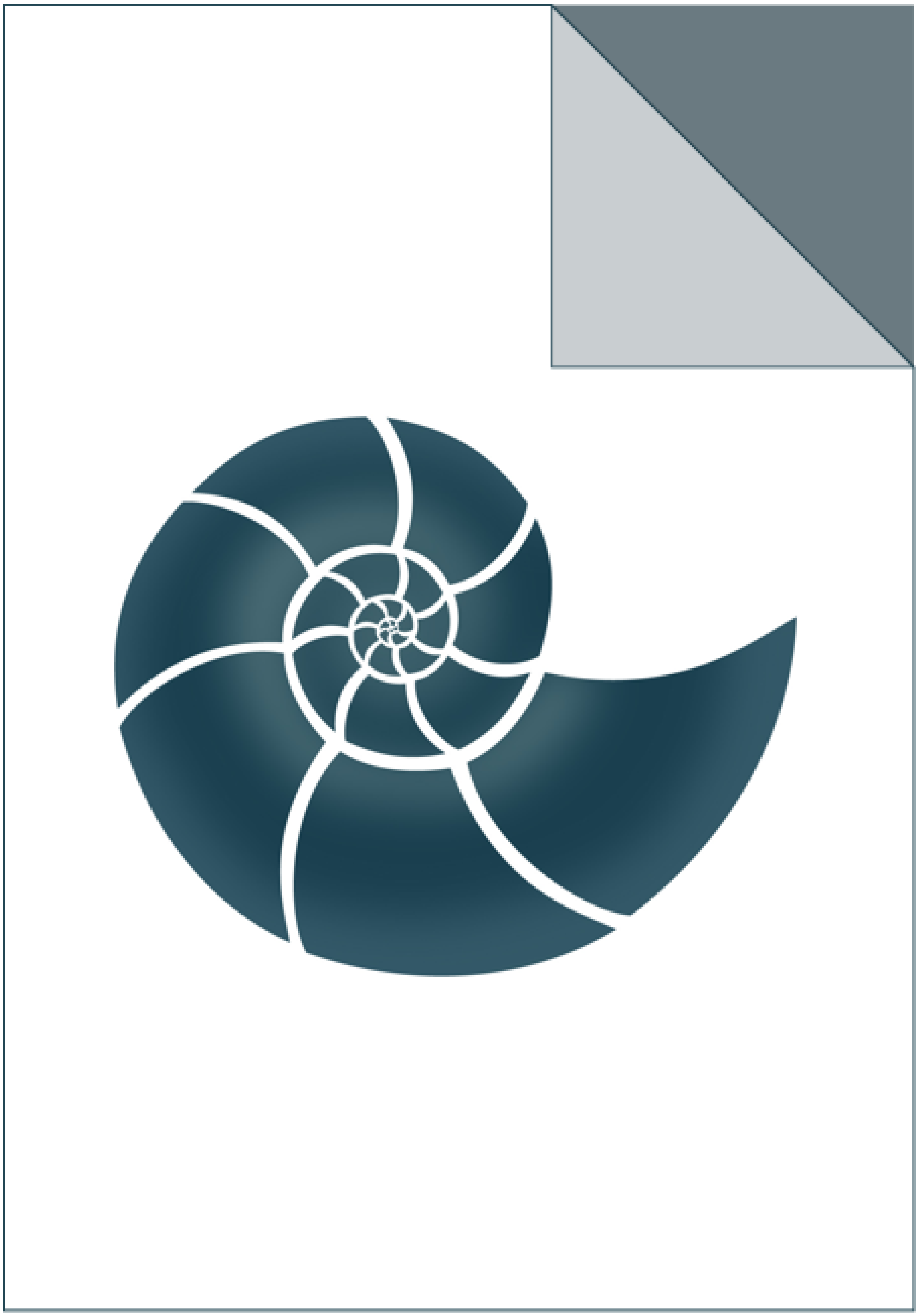
A simple example shows how to use UnionFind algorithm.

Keywords:

- no_keywords

Categories:

- no_categories



5.2.59 ex_WebServer

Simple test for WebServer class

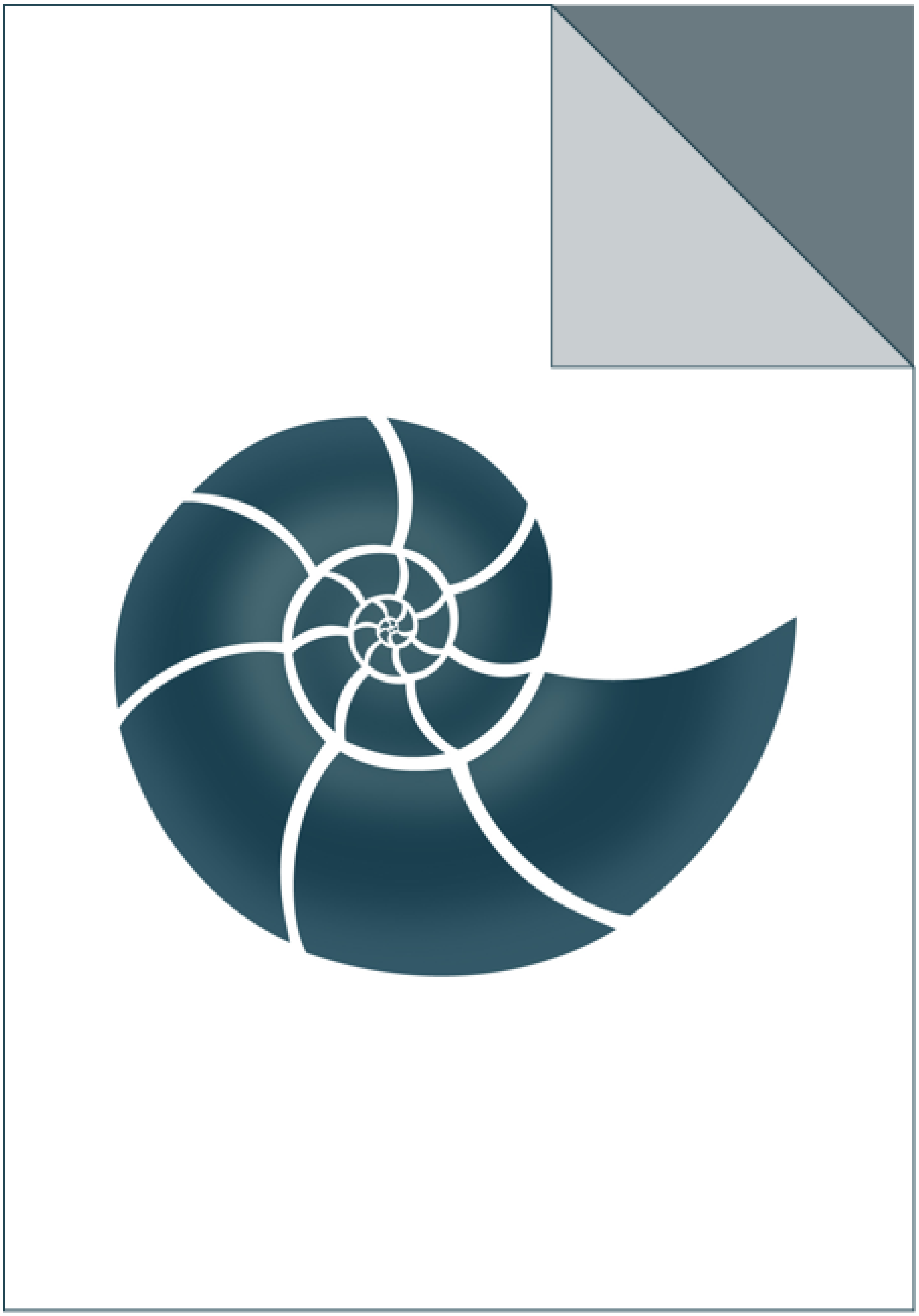
USAGE: ex_WebServer [port]

Keywords:

- no_keywords

Categories:

- no_categories



5.2.60 ex_XML

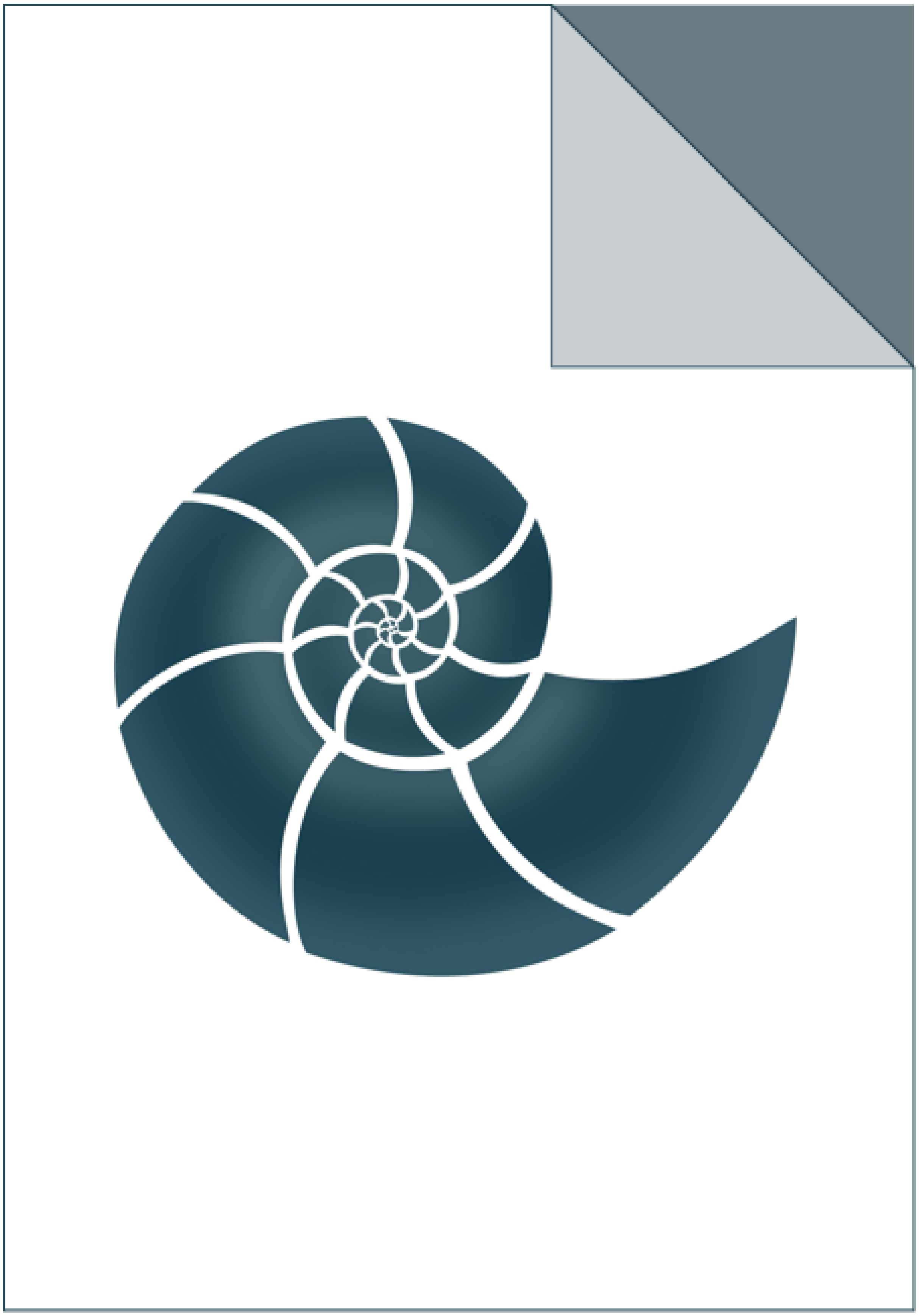
Simple for XML I/O utils.

Keywords:

- no_keywords

Categories:

- no_categories



5.2.61 ex_alignment_io

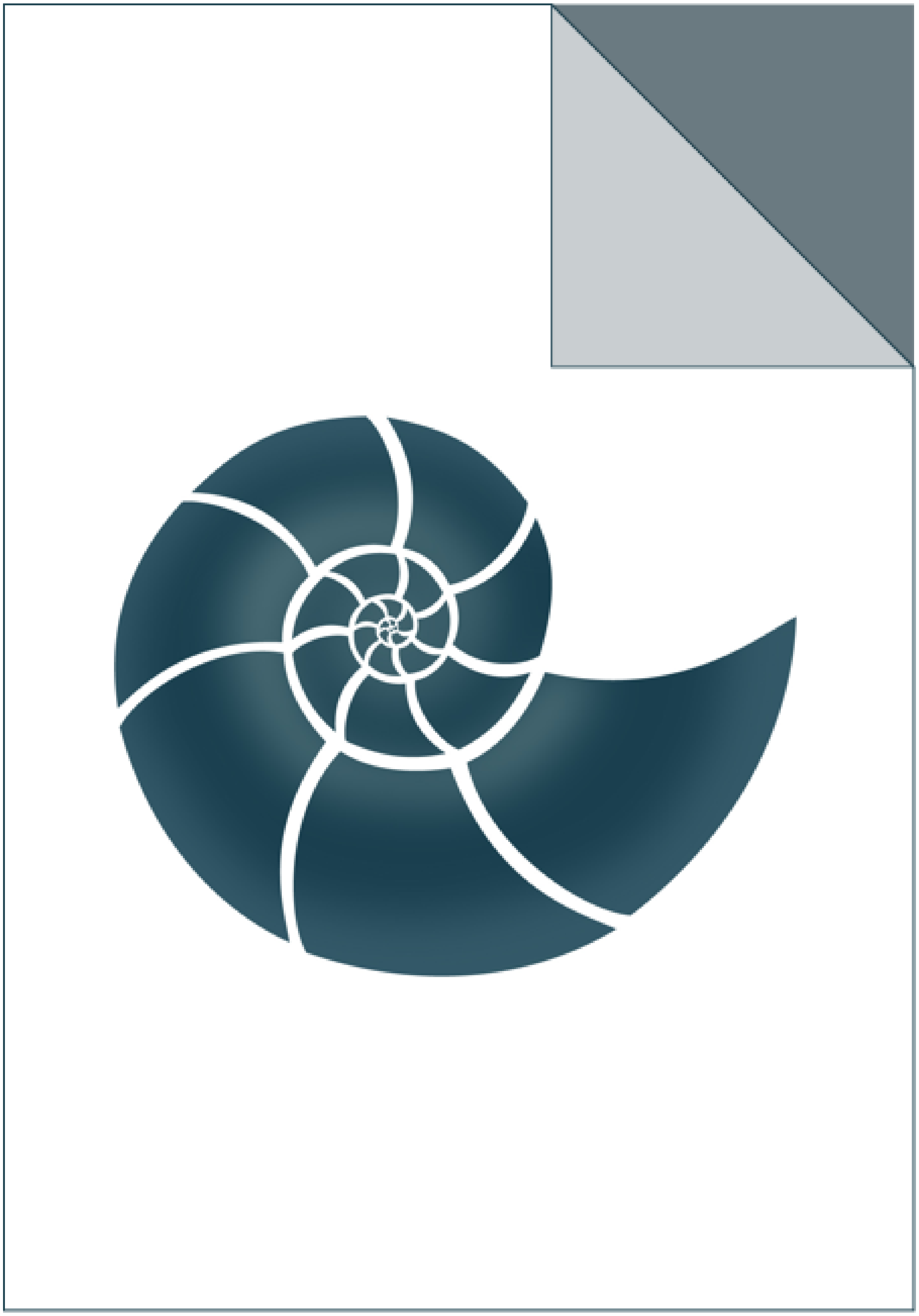
Read alignment in Edinburgh format or calculate a new one from given sequences; write Edinburgh.

Keywords:

- no_keywords

Categories:

- no_categories



5.2.62 ex_basic_algebra

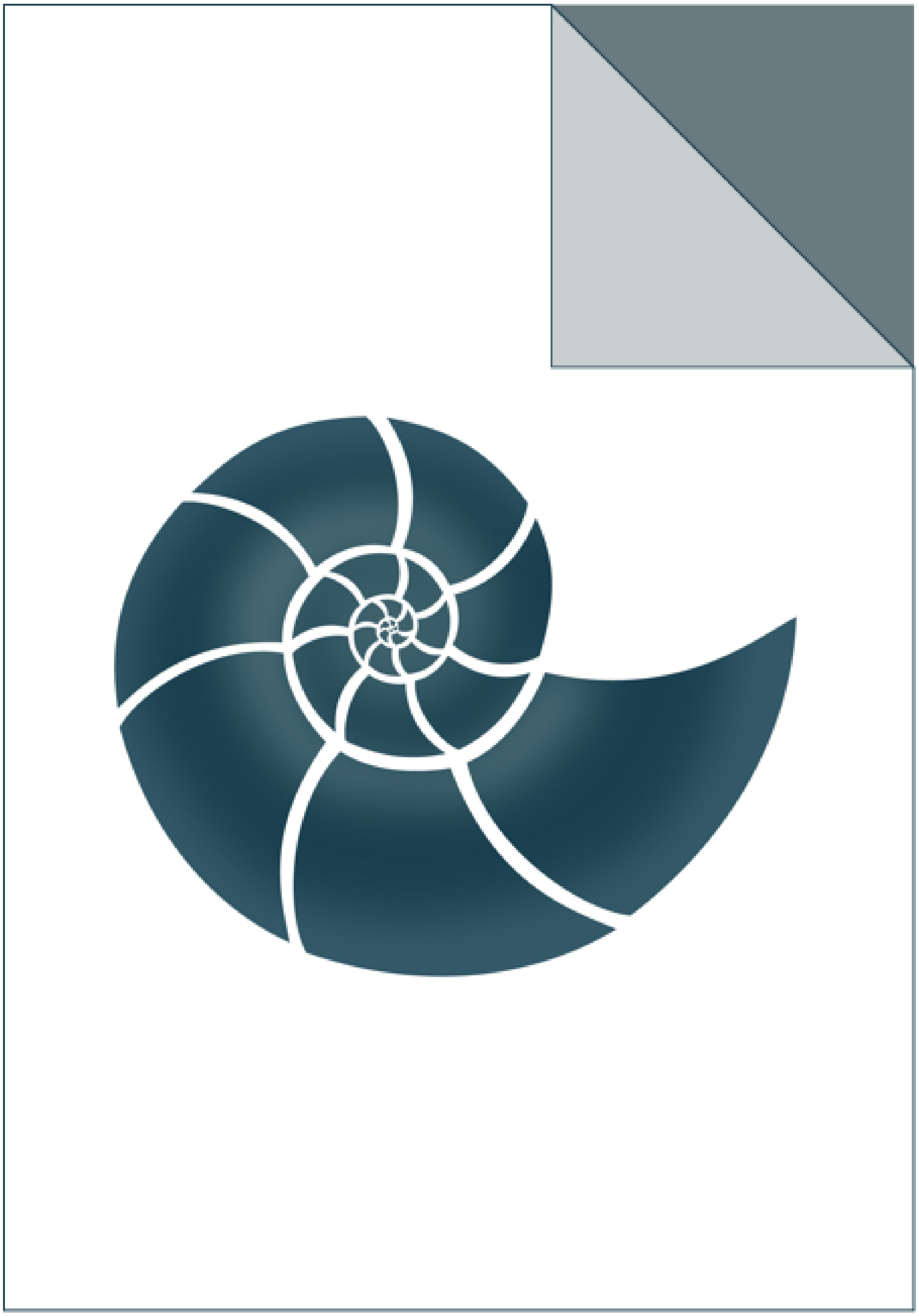
ex_basic_algebra illustrates how to calculate eigenvalues and eigenvectors for a 3x3 matrix

Keywords:

- no_keywords

Categories:

- no_categories



5.2.63 ex_benchmark_quick_seq_identity

ex_benchmark_quick_seq_identity estimates sequence identity without actual aligning the sequences. The purpose of this

program is to compare three different implementations of the method (BitSet should be much faster than tuple counting)

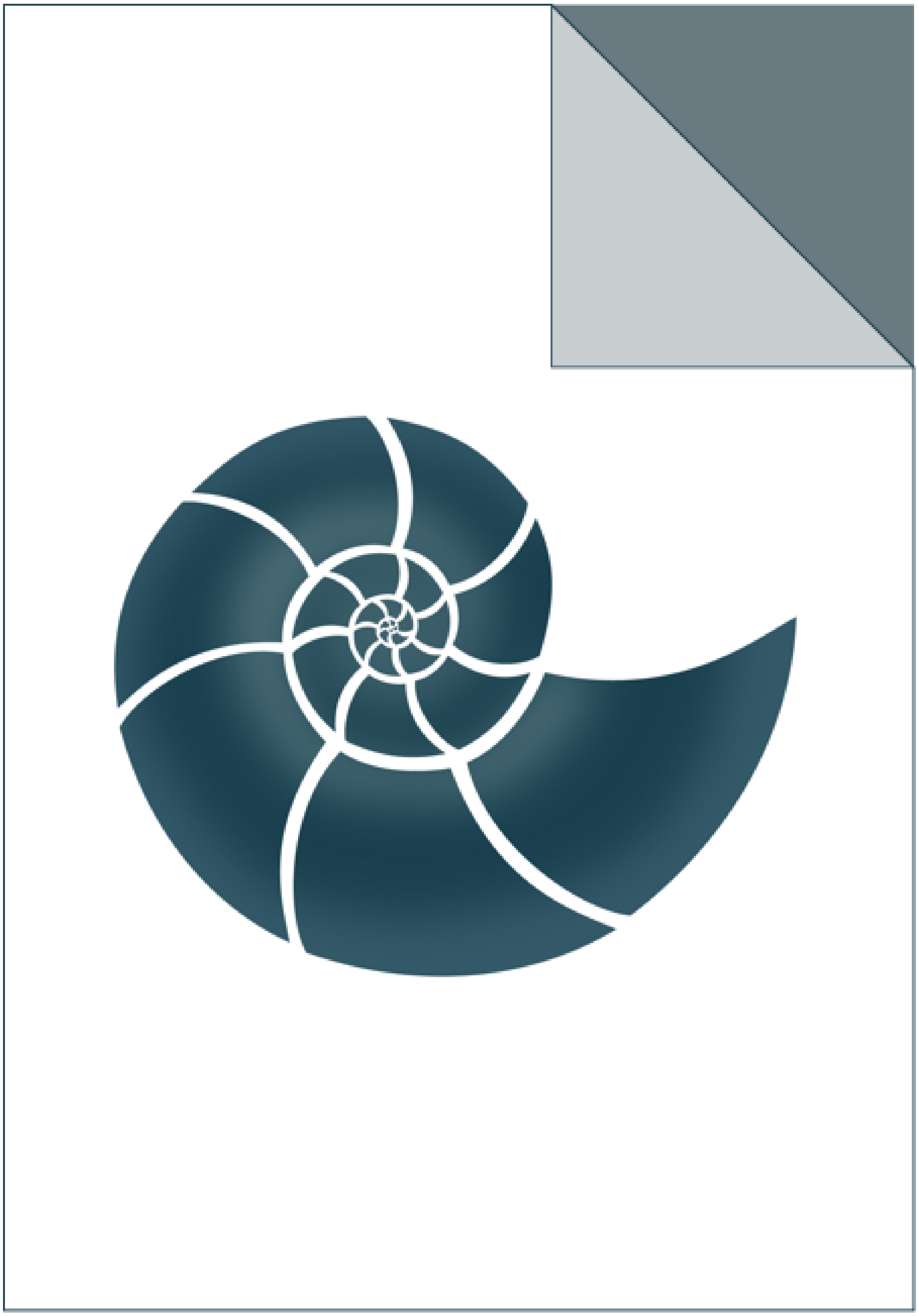
USAGE: ./ex_benchmark_quick_seq_identity cyped.CYP109.fasta

Keywords:

- no_keywords

Categories:

- no_categories



5.2.64 ex_chi2_independence_test

Performs chi-square test: calculates p-value for a given number of DOFs. Alternatively,

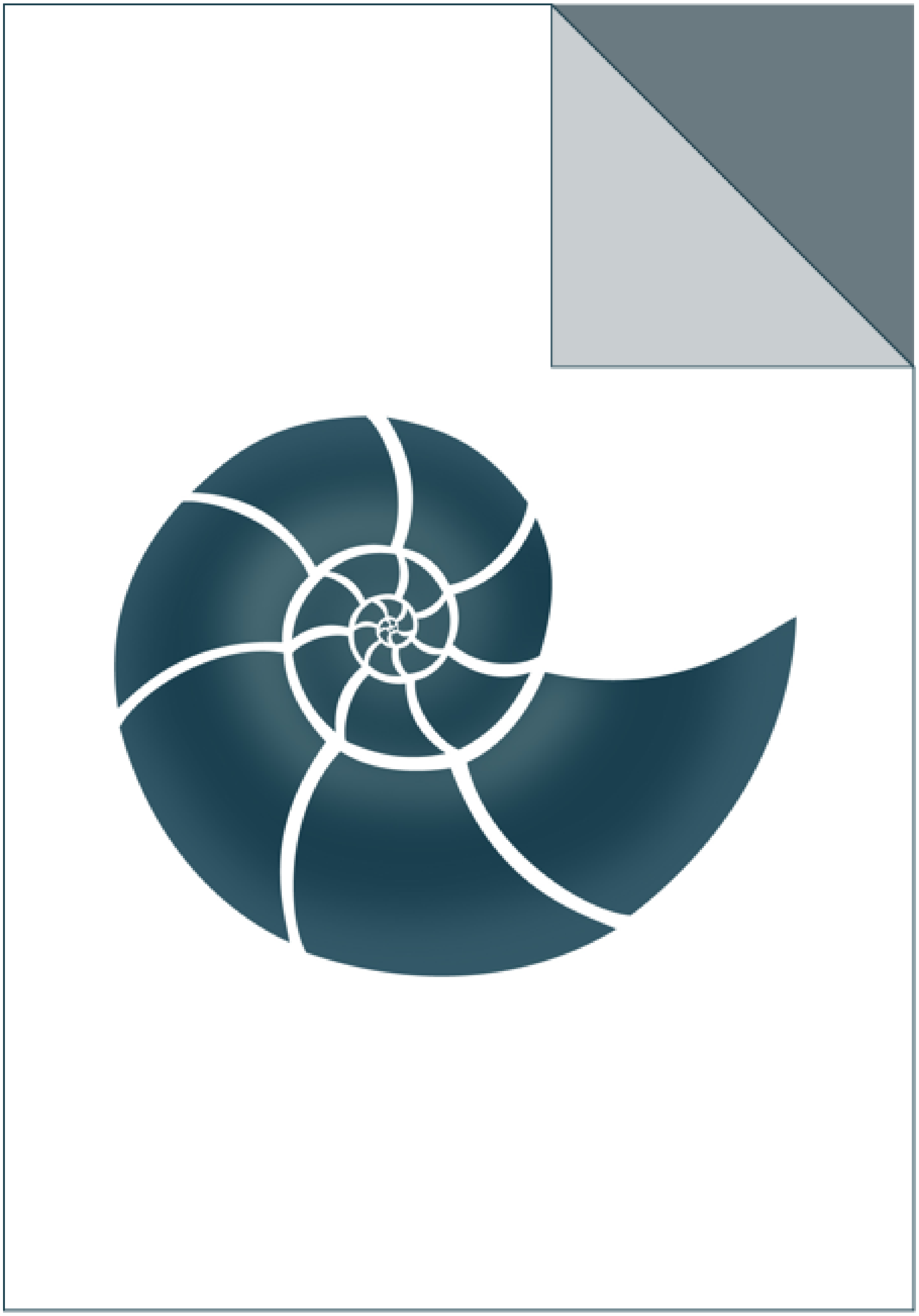
it can read a contingency matrix from a file and calcutate test for independence of its two first rows When no input data is provided, the example performs Chi-square independence test on a test data USAGE: ex_chi2_independence_test [n_dofs chi2_value] ex_chi2_independence_test [input_contingency_matrix_file

Keywords:

- no_keywords

Categories:

- no_categories



5.2.65 ex_consecutive_find

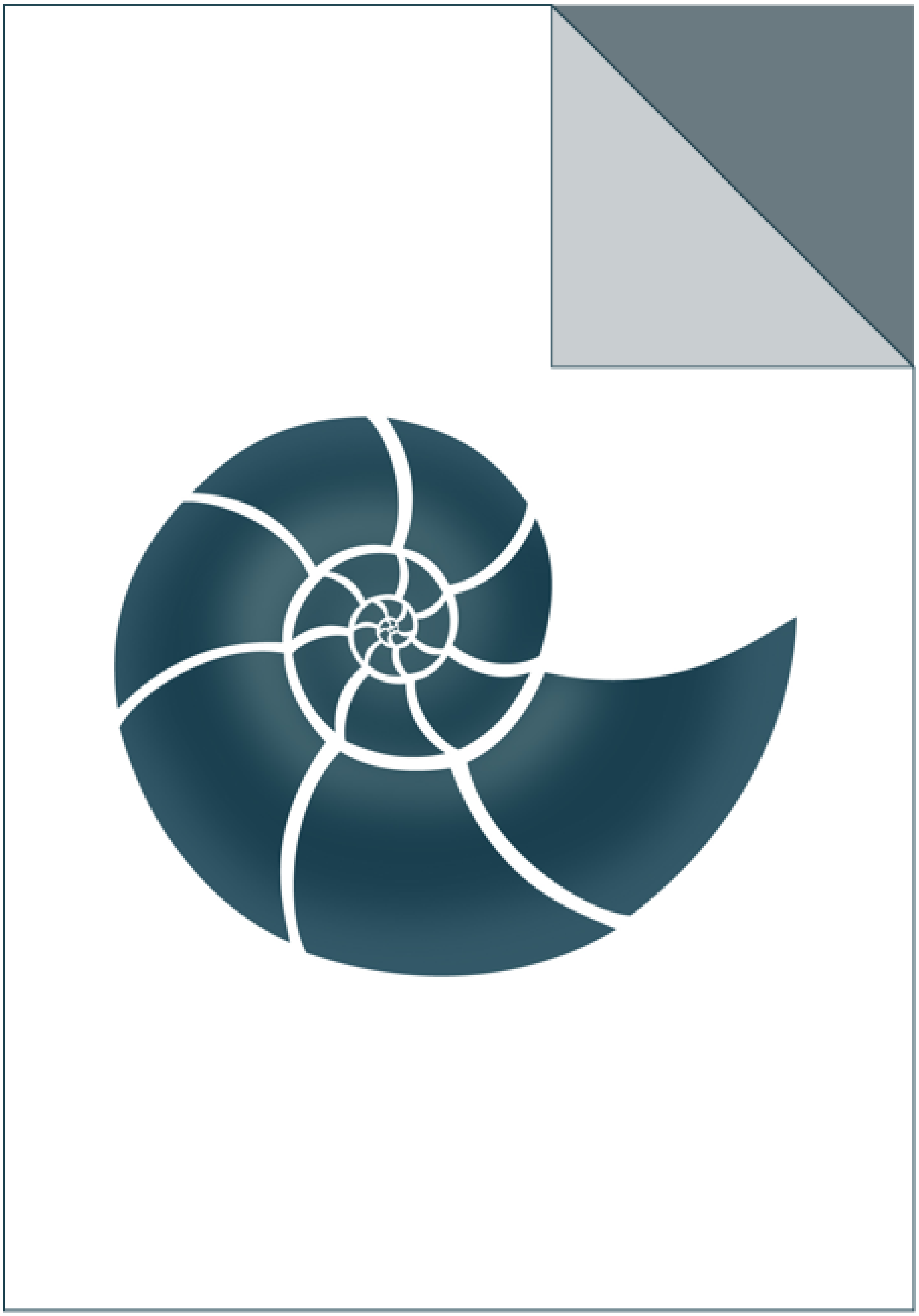
Shows how to find islands of consecutive elements in a container

Keywords:

- no_keywords

Categories:

- no_categories



5.2.66 ex_count_residues_by_type

ex_count_residues_by_type reads a Multiple Sequence Alignment (MSA) in ClustalW format and counts residues by its type

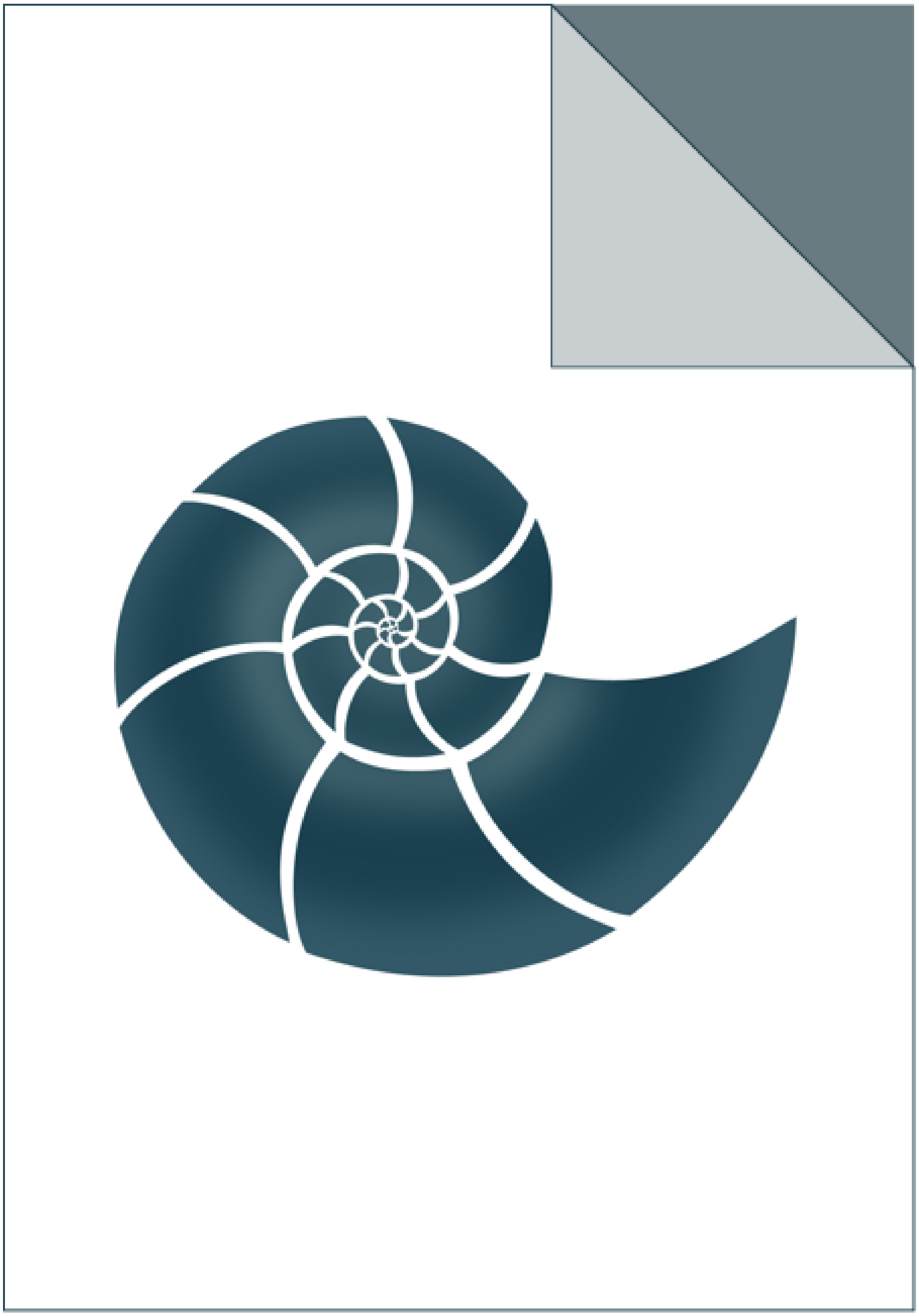
USAGE: ./ex_count_residues_by_type cyped.CYP109.aln

Keywords:

- no_keywords

Categories:

- no_categories



5.2.67 ex_define_rotamer

ex_define_rotamer prints rotamer type (M-P-T code) for each amino acid

residue in the input PDB structure USAGE: ex_define_rotamer 5edw.pdb

Keywords:

- no_keywords

Categories:

- no_categories

5.2.68 ex_expectation_maximization

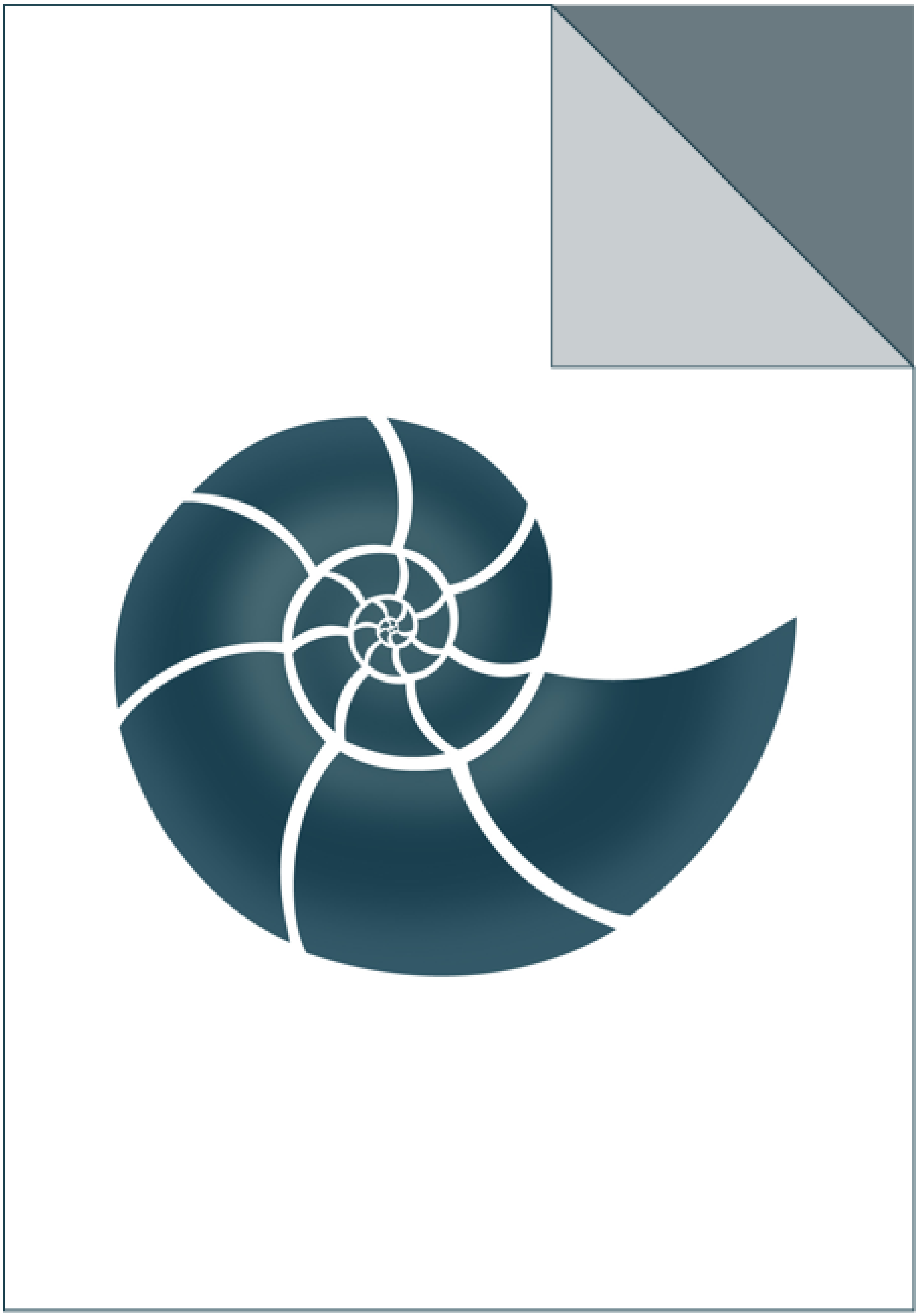
Example showing how to use expectation-maximization method

Keywords:

- no_keywords

Categories:

- no_categories



5.2.69 ex_find_side_group

Reads a PDB file and prints names of all atoms in residue side chains

USAGE: ex_find_side_group 2gb1.pdb

Keywords:

- no_keywords

Categories:

- no_categories

5.2.70 ex_goodman_kruskal_rank_correlation

> The program read a contingency matrix from a file and calculates Goodman and Kruskal's gamma parameters

which is a measure of rank correlation.

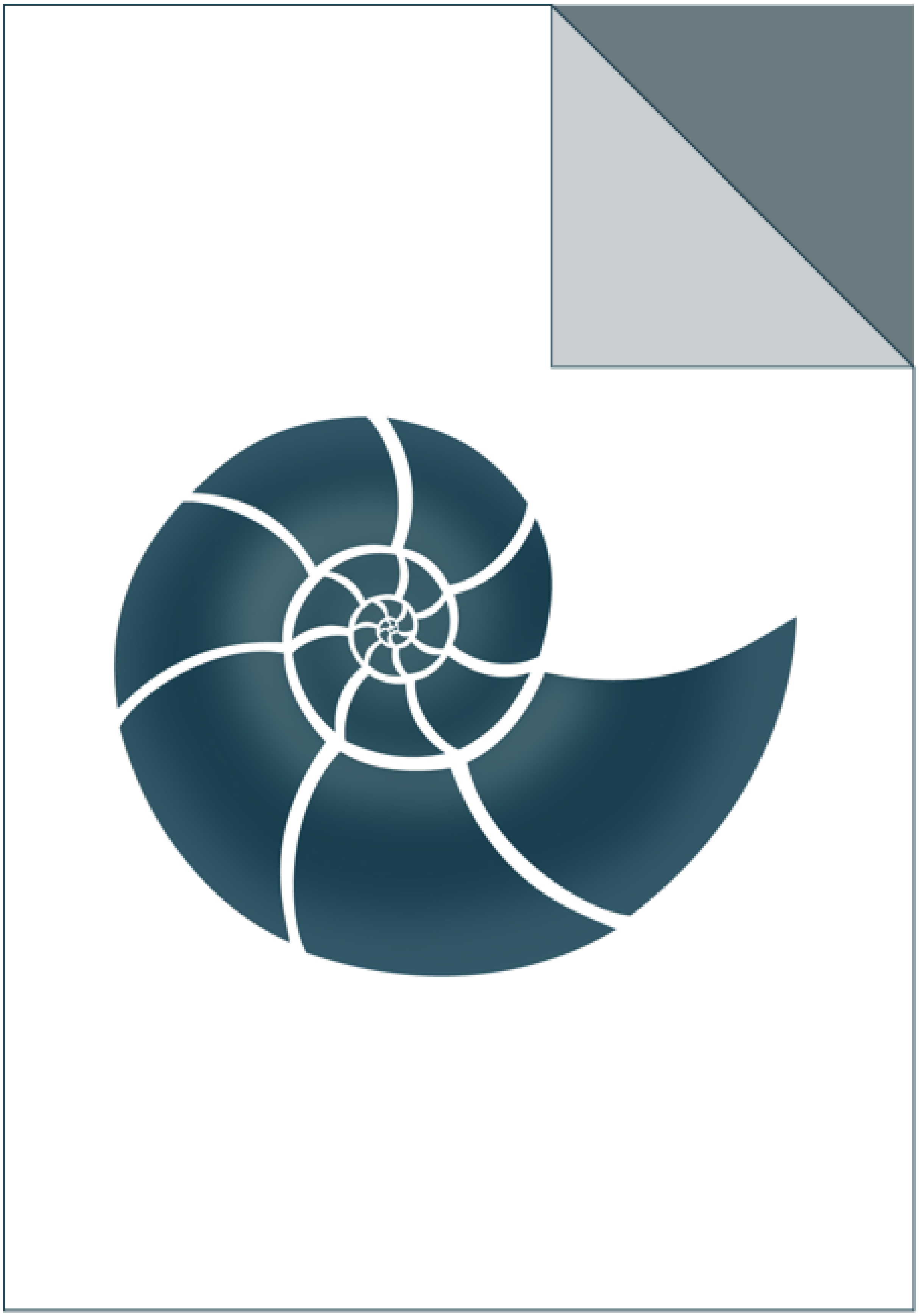
USAGE: ex_goodman_kruskal_rank_correlation input_contingency_matrix_fil

Keywords:

- no_keywords

Categories:

- no_categories



5.2.71 ex_greedy_clustering

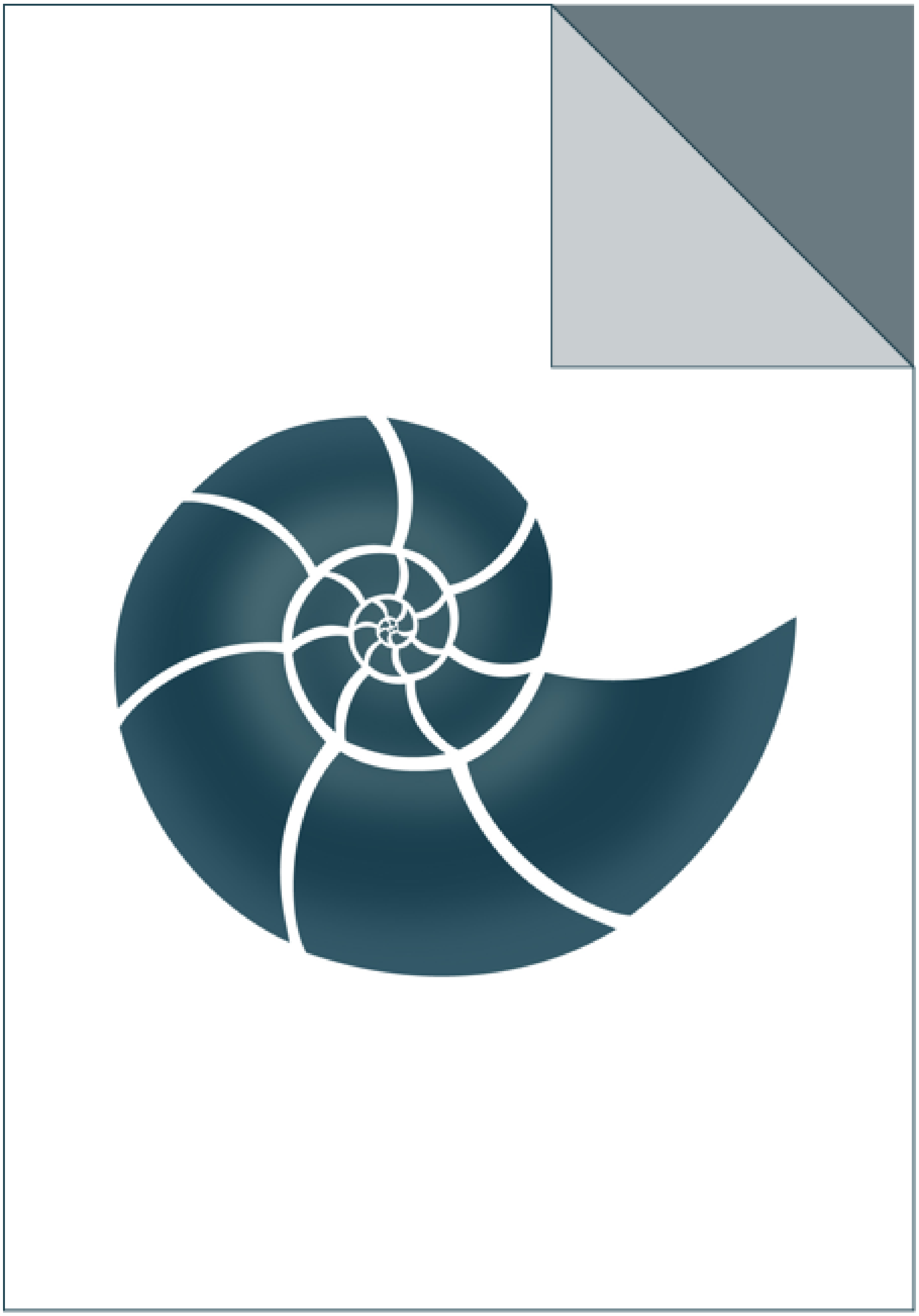
Example showing how to use greedy clustering method.

Keywords:

- no_keywords

Categories:

- no_categories



5.2.72 ex_intersect_sorted

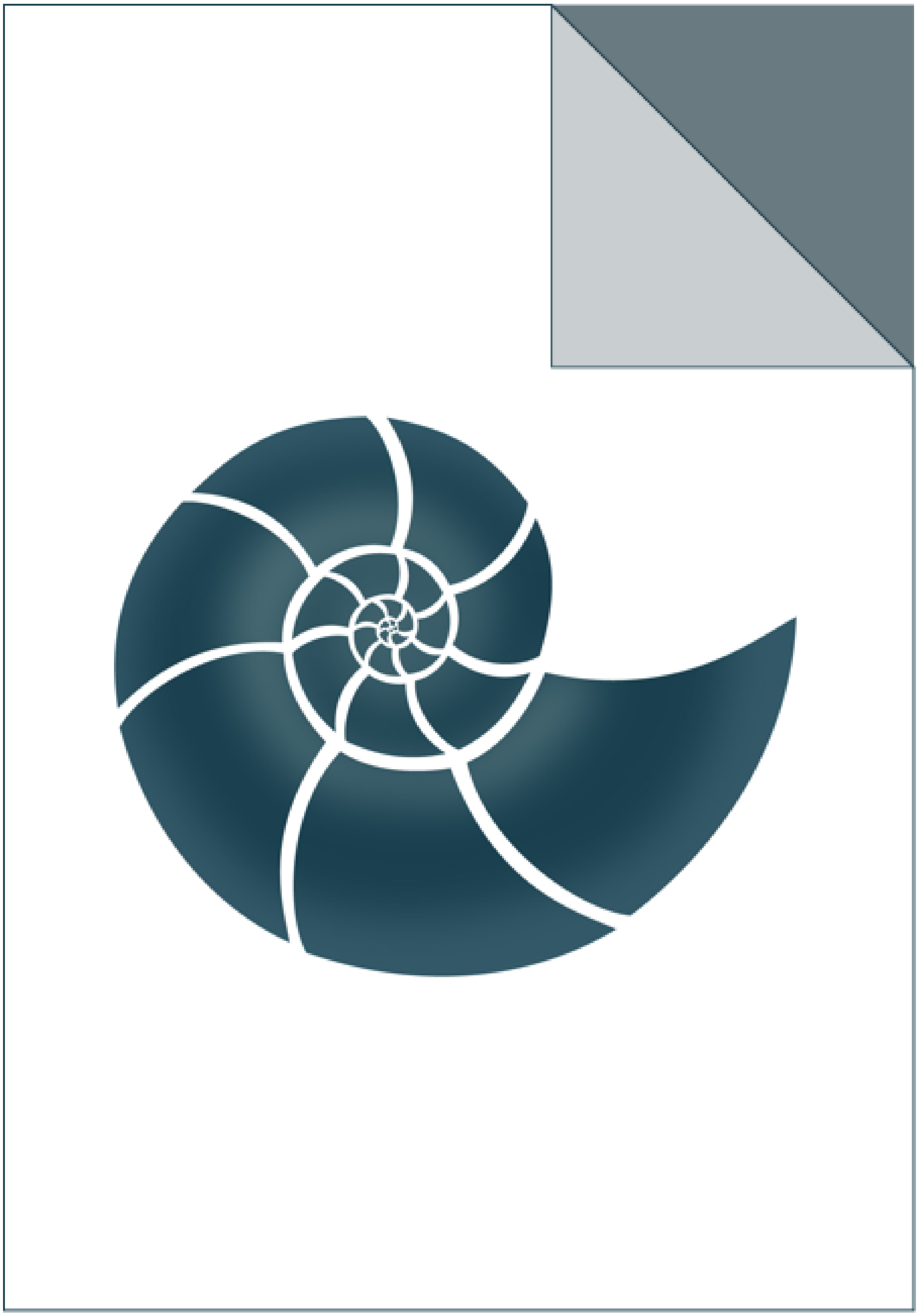
Shows how to find an intersection of two sorted vectors of data

Keywords:

- no_keywords

Categories:

- no_categories



5.2.73 ex_k_tuples

Reads a set of sequences given in a FASTA format and prints all 4-tuples that can be created from it.

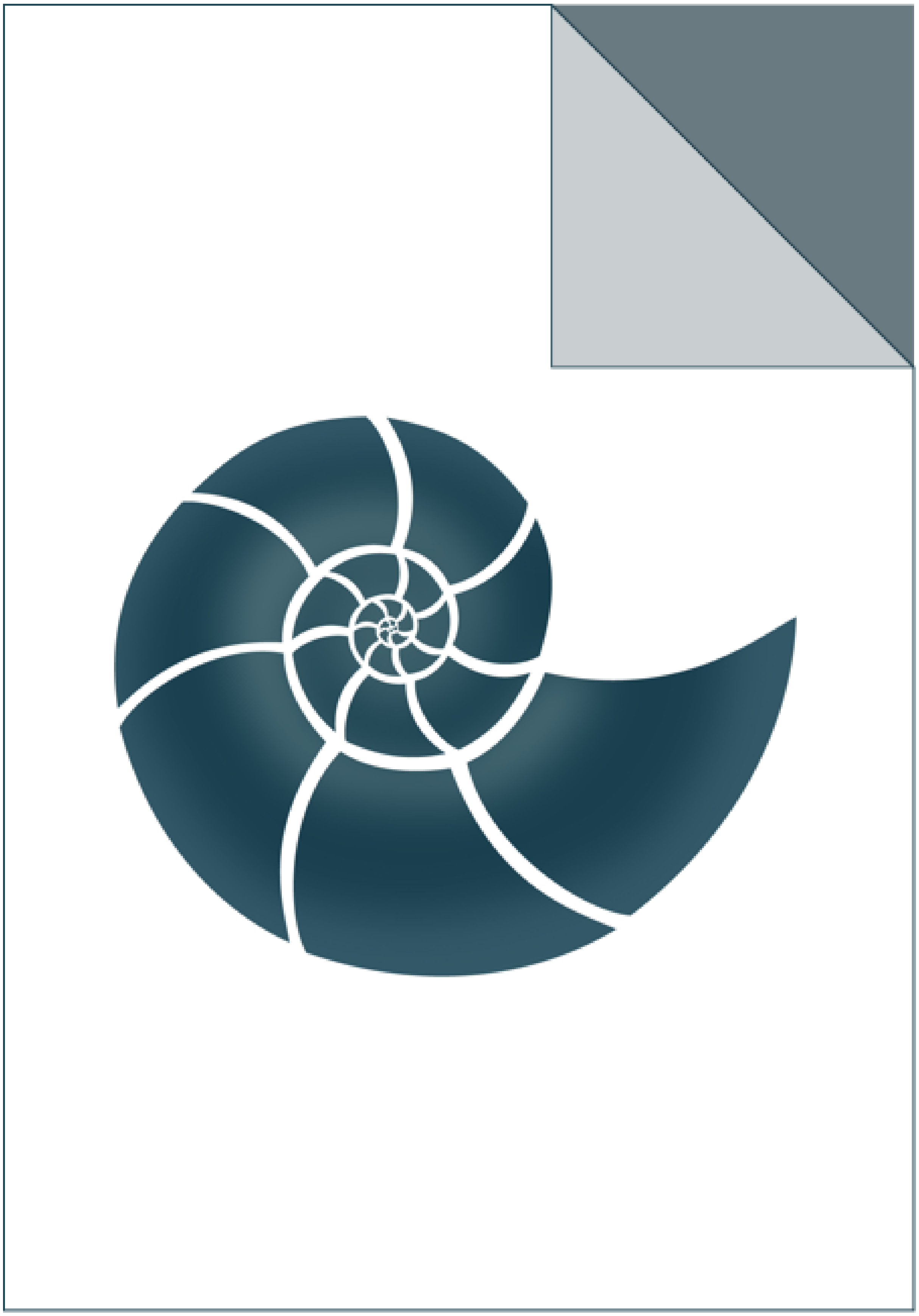
USAGE: ex_k_tuples sequences.fasta [alphabet_name]

Keywords:

- no_keywords

Categories:

- no_categories



5.2.74 ex_local_BBQ_coordinates

ex_local_BBQ_coordinates reads a PDB file and prints local coordinates for sidechain atoms

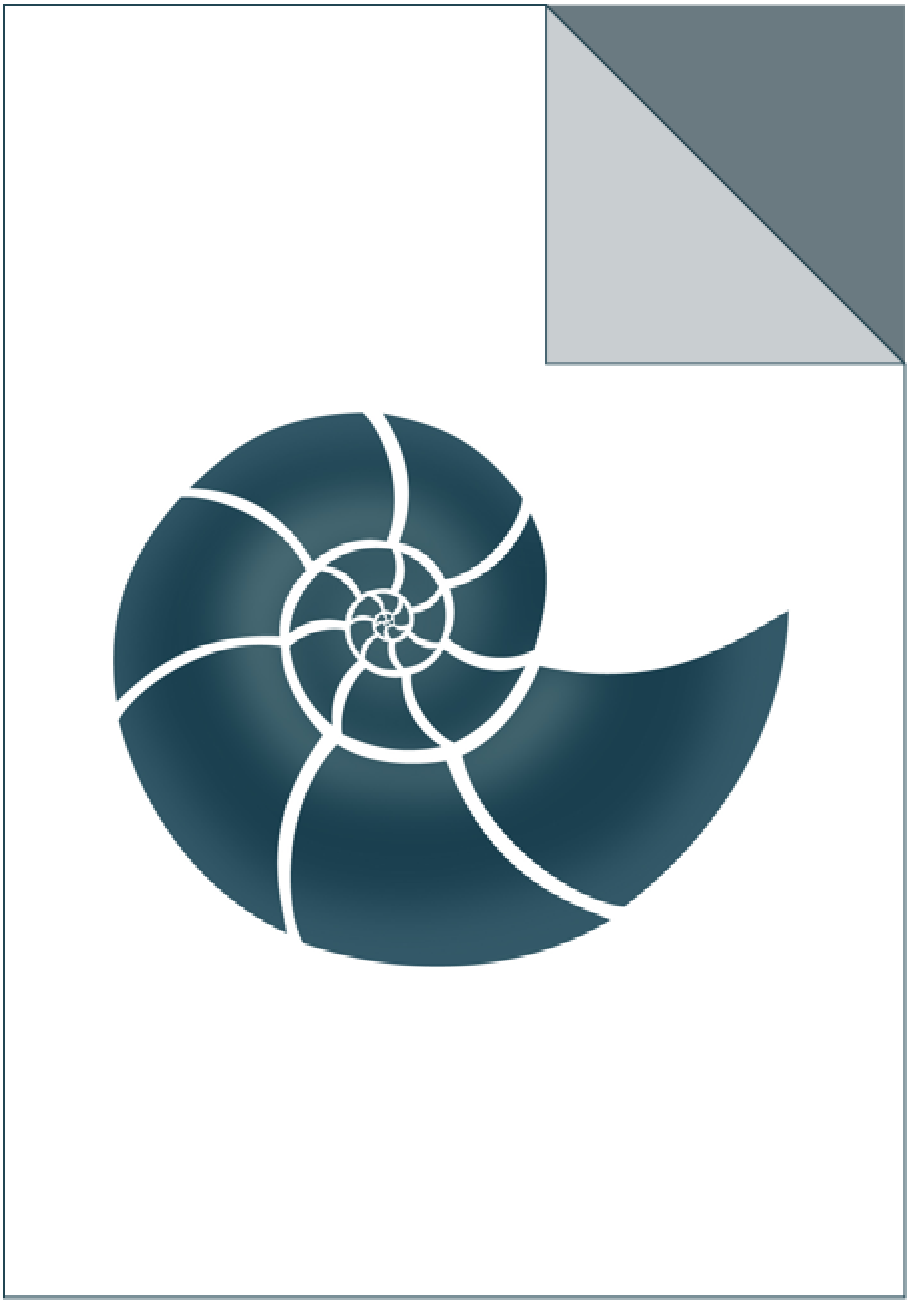
USAGE: ex_local_BBQ_coordinates 5edw.pdb

Keywords:

- no_keywords

Categories:

- no_categories



5.2.75 ex_local_coordinates_three_atoms

ex_local_coordinates_three_atoms reads a PDB file and prints local coordinates for sidechain atoms.

For every residue, a local coordinate system (LCS) is constructed based on N, C-alpha and C atoms. The program prints positions of all atoms of a residue in its LCS

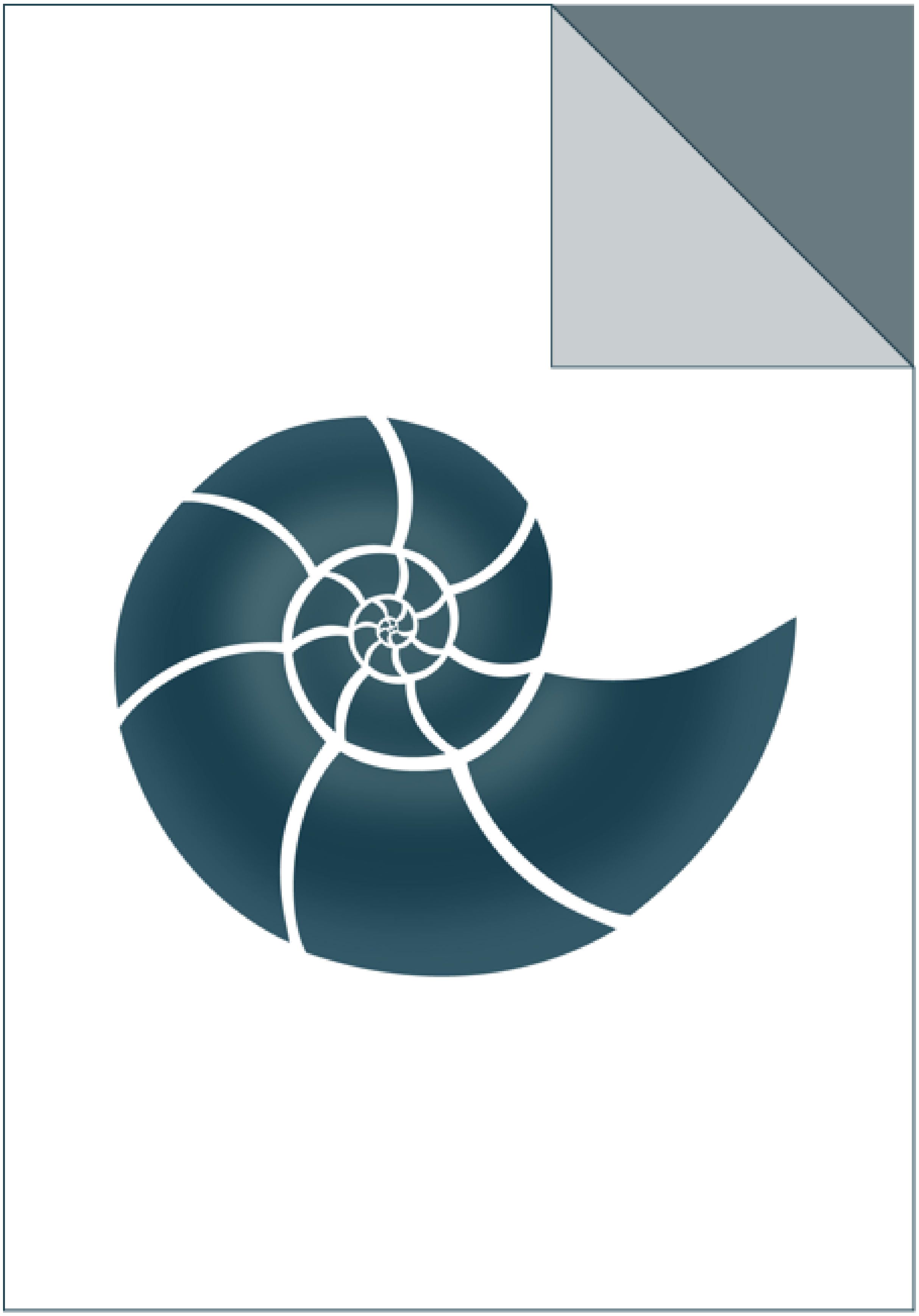
USAGE: ex_local_coordinates_three_atoms 5edw.pdb

Keywords:

- no_keywords

Categories:

- no_categories



5.2.76 ex_monomer_io

The program converts a monomer structure from CIF format to internal formats used by BioShell.

Use it to register your own monomer which is missing in BioShell library. The program is also used to create 'monomers.txt' file from BioShell distribution (located in ./data/ directory). In order to do so, download the fresh repository of monomers in CIF format from:

http://ligand-expo.rcsb.org/dictionaries/Components-pub.cif

and run the program. Then replace the released monomers.txt file with the new one

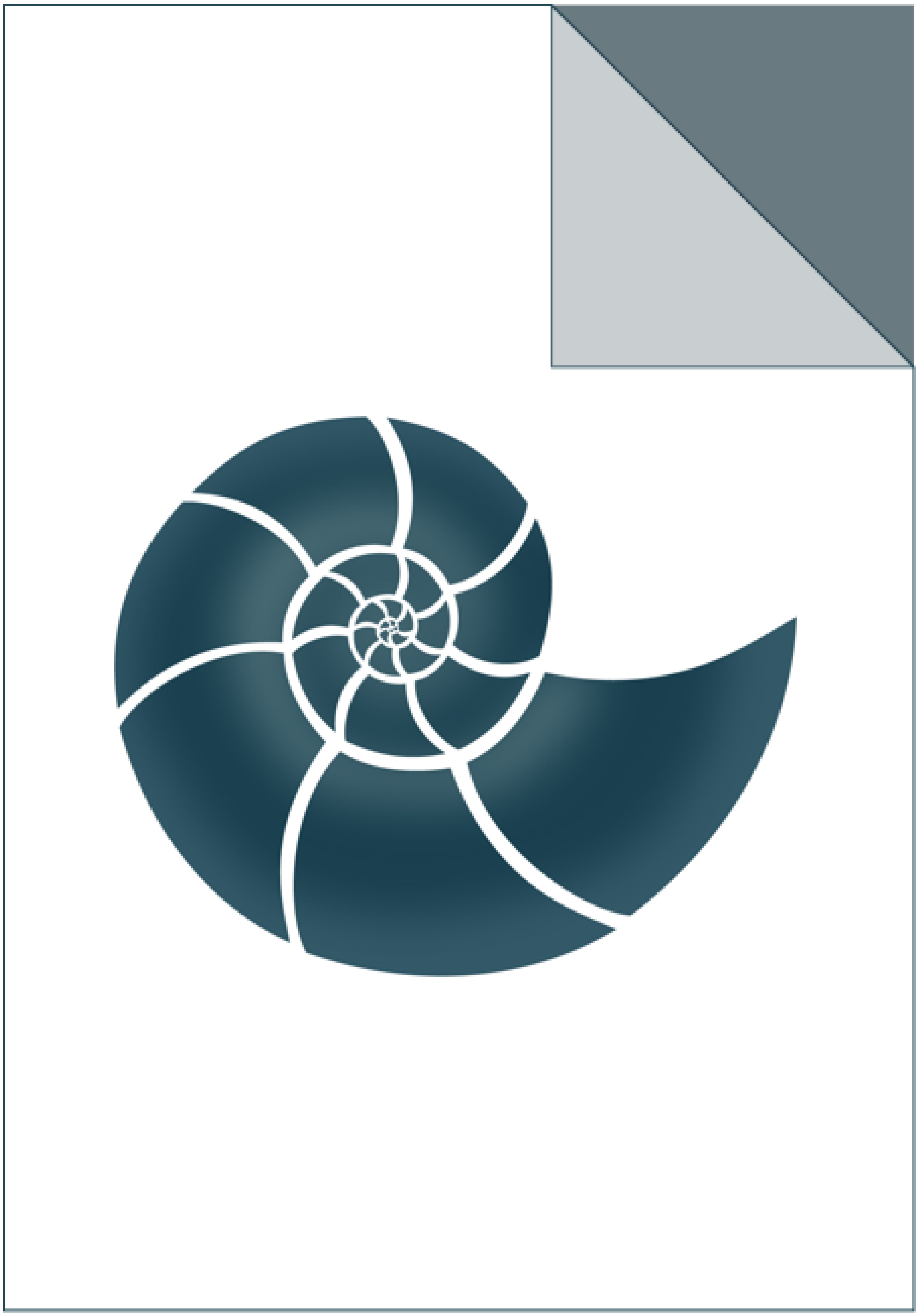
USAGE: ./ex_monomer_io -in::monomers::cif=HEM.cif -out:file=hem.txt ./ex_monomer_io -in::monomers::cif=Components-pub.cif

Keywords:

- no_keywords

Categories:

- no_categories



5.2.77 ex_pdb_to_fasta

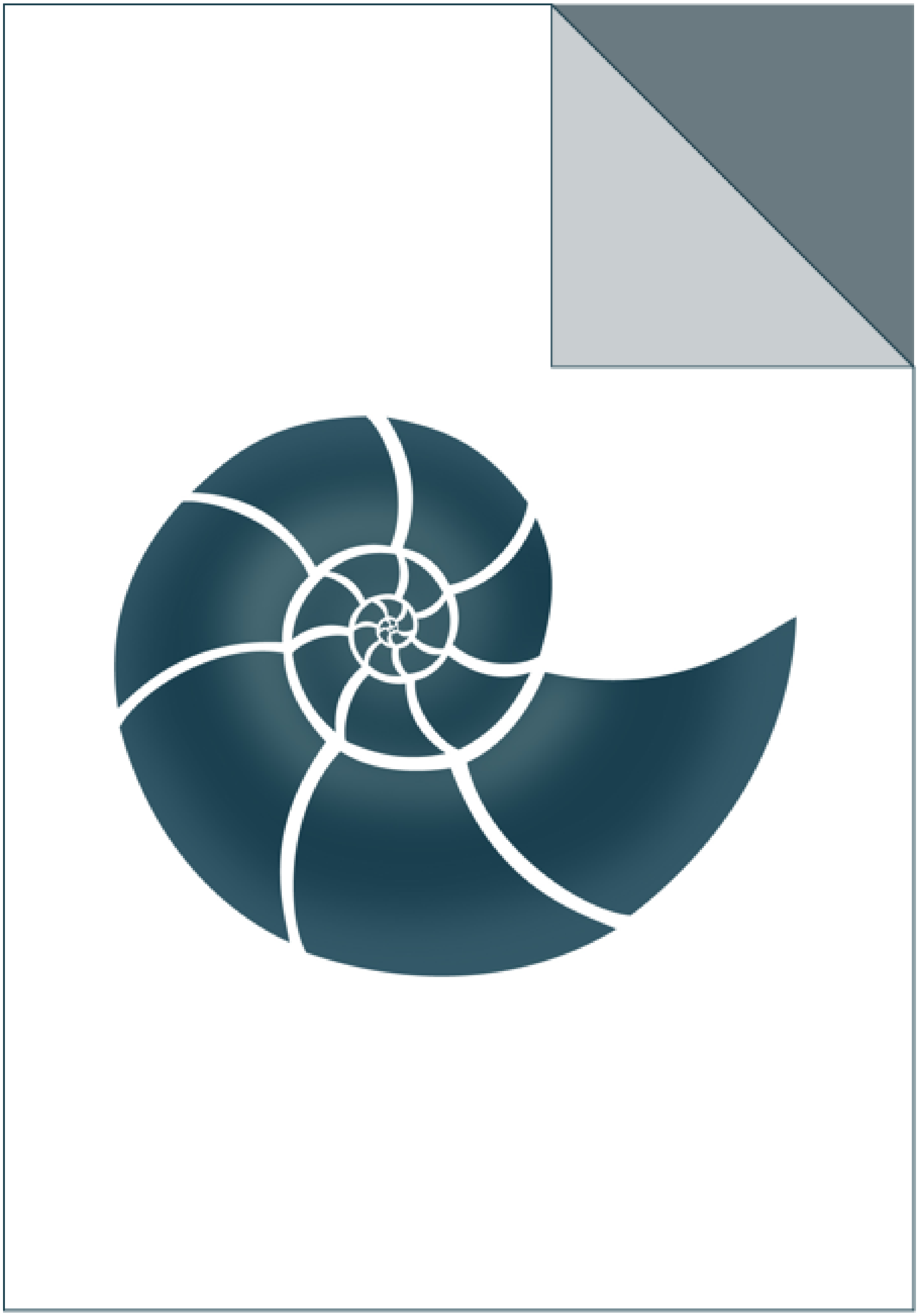
Reads a PDB file and writes protein sequence(s) in FASTA format.

Keywords:

- no_keywords

Categories:

- no_categories



5.2.78 ex_peptide_hydrogen

ex_peptide_hydrogen reconstructs peptide hydrogen atoms using BioShell algorithm,

where amide H is placed in reference to its N atom. Resulting coordinates are printed on the screen. The program also computes the amide-H positions using DSSP approach and calculates the average error (in Angstroms) between the two methods.

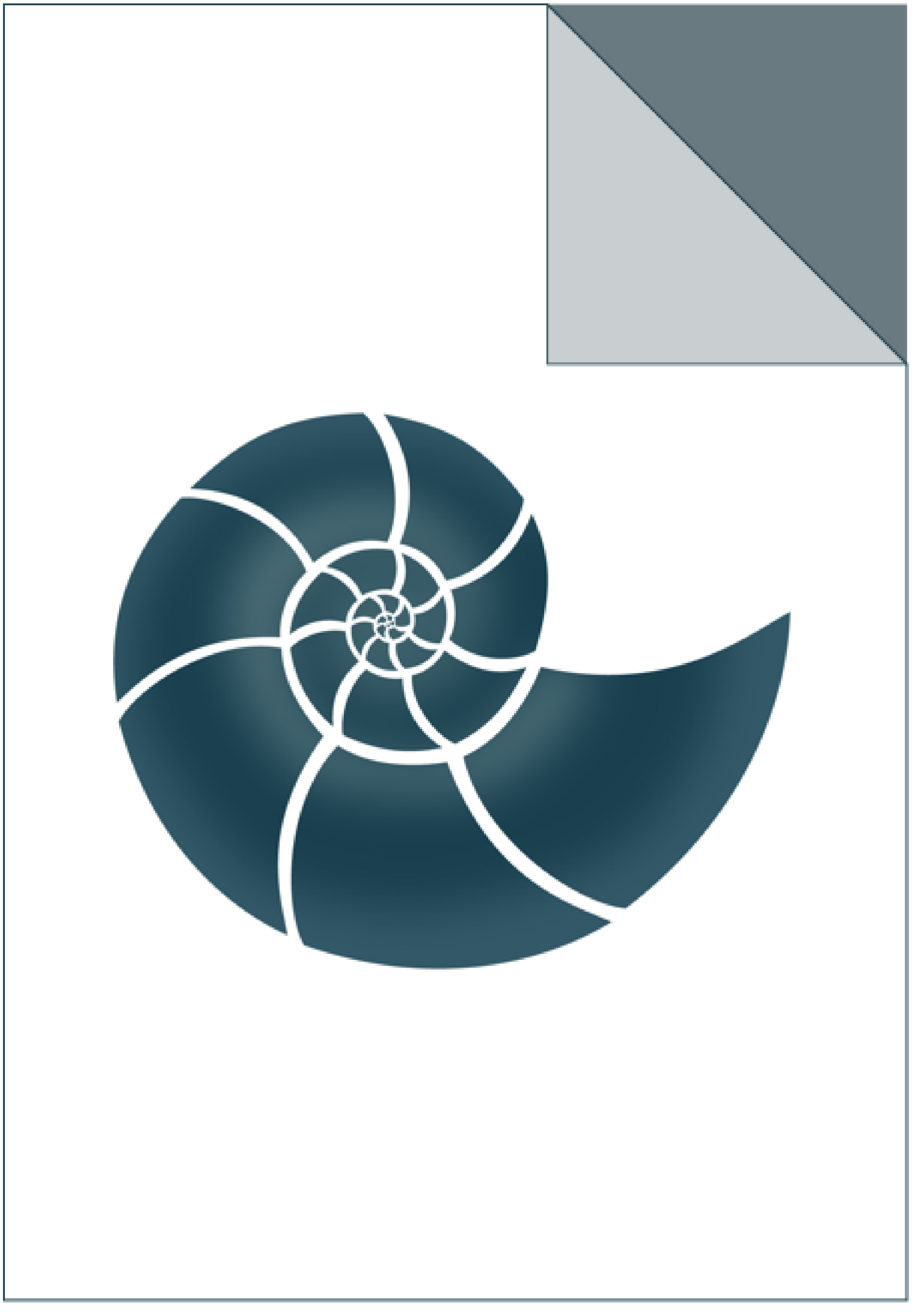
USAGE: ex_peptide_hydrogen 5edw.pdb

Keywords:

- no_keywords

Categories:

- no_categories



5.2.79 ex_protein_peptide_interface

ex_protein_peptide_interface finds atomic contacts between a receptor and a peptide found in an input PDB file.

Its output provides: protein residue name and ID, protein chain ID, peptide protein name and ID, peptide chain ID, minimum distance between the residues, e.g.

ILE 36 A ARG 104 X 5.92977 LEU 44 A ARG 104 X 5.92685 LEU 44 A LEU 108 X 5.57779 GLU 45 A THR 102 X 6.81994

USAGE: ex_protein_peptide_interface 1dt7.pdb 7.0

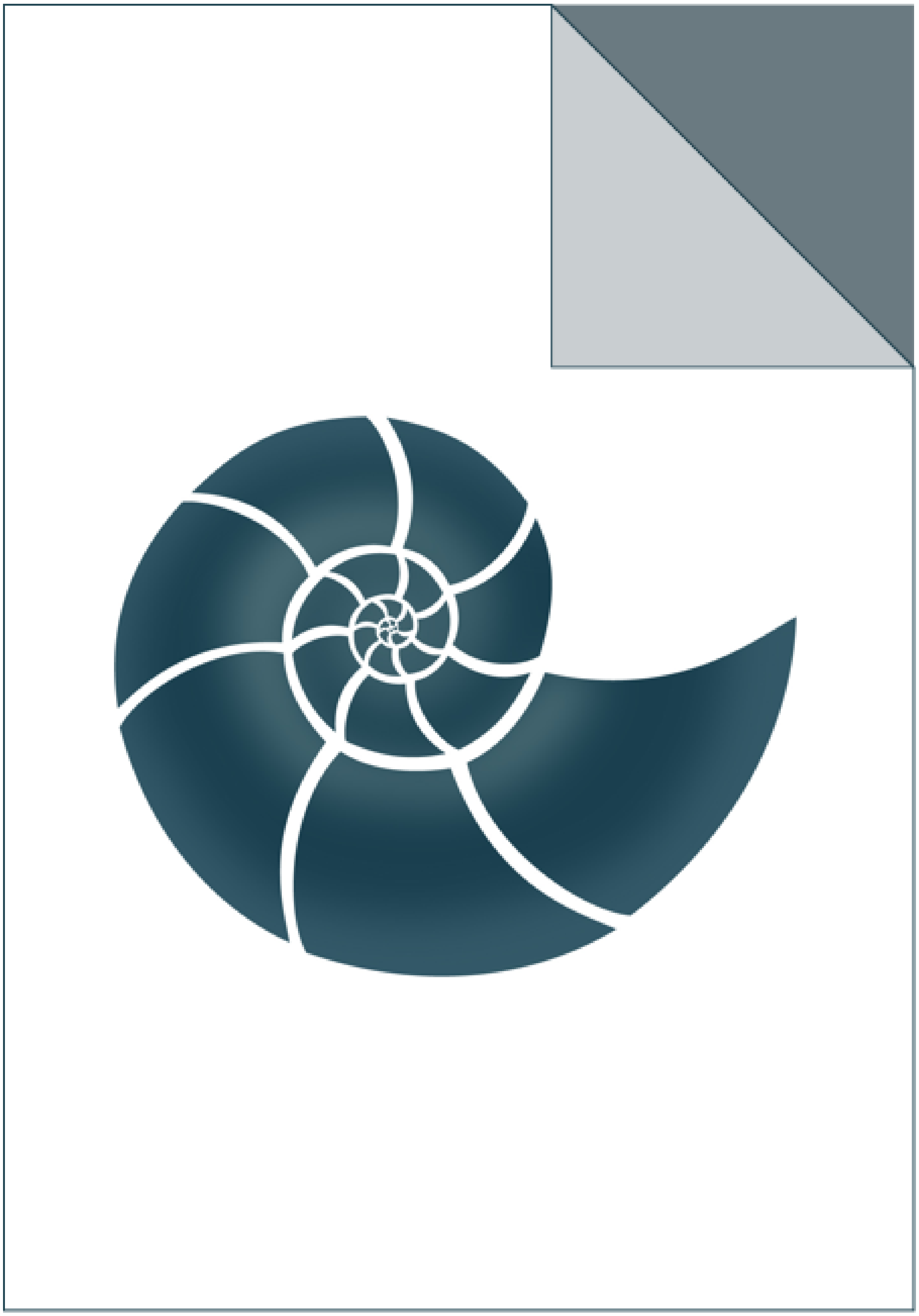
where 1dt7.pdb id an input file and 7.0 - contact distance in Angstroms

Keywords:

- no_keywords

Categories:

- no_categories



5.2.80 ex_read_properties_file

Simple test for ex_read_properties_file function

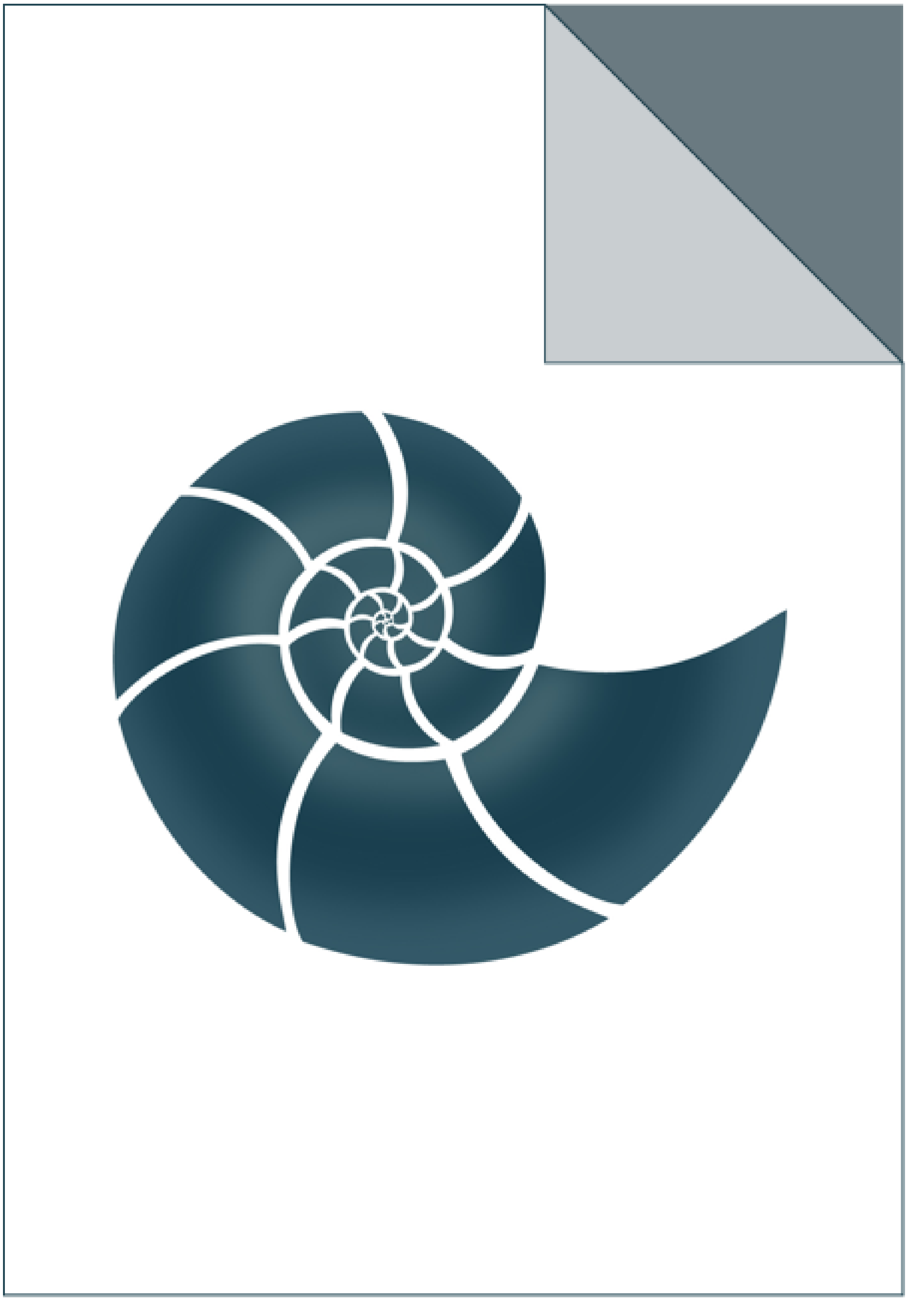
USAGE: ex_read_properties_file input_file.properties

Keywords:

- no_keywords

Categories:

- no_categories



5.2.81 ex_selection_protocols

ex_selection_protocols shows how to use selection protocols

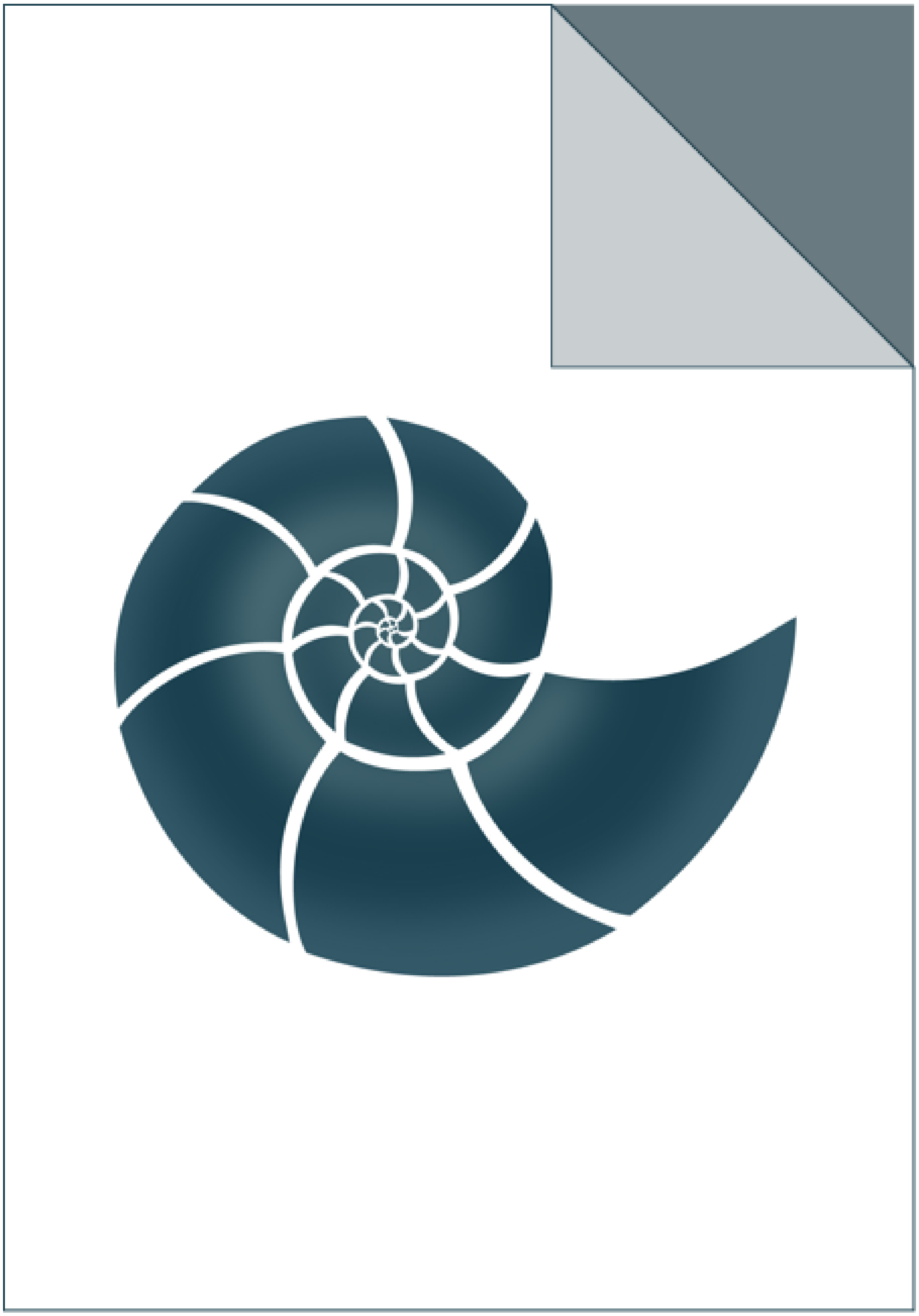
USAGE: ex_selection_protocols 5edw.pdb

Keywords:

- no_keywords

Categories:

- no_categories



5.2.82 ex_seq_io

ex_seq_io reads a SEQ file and prints its contents in FASTA format

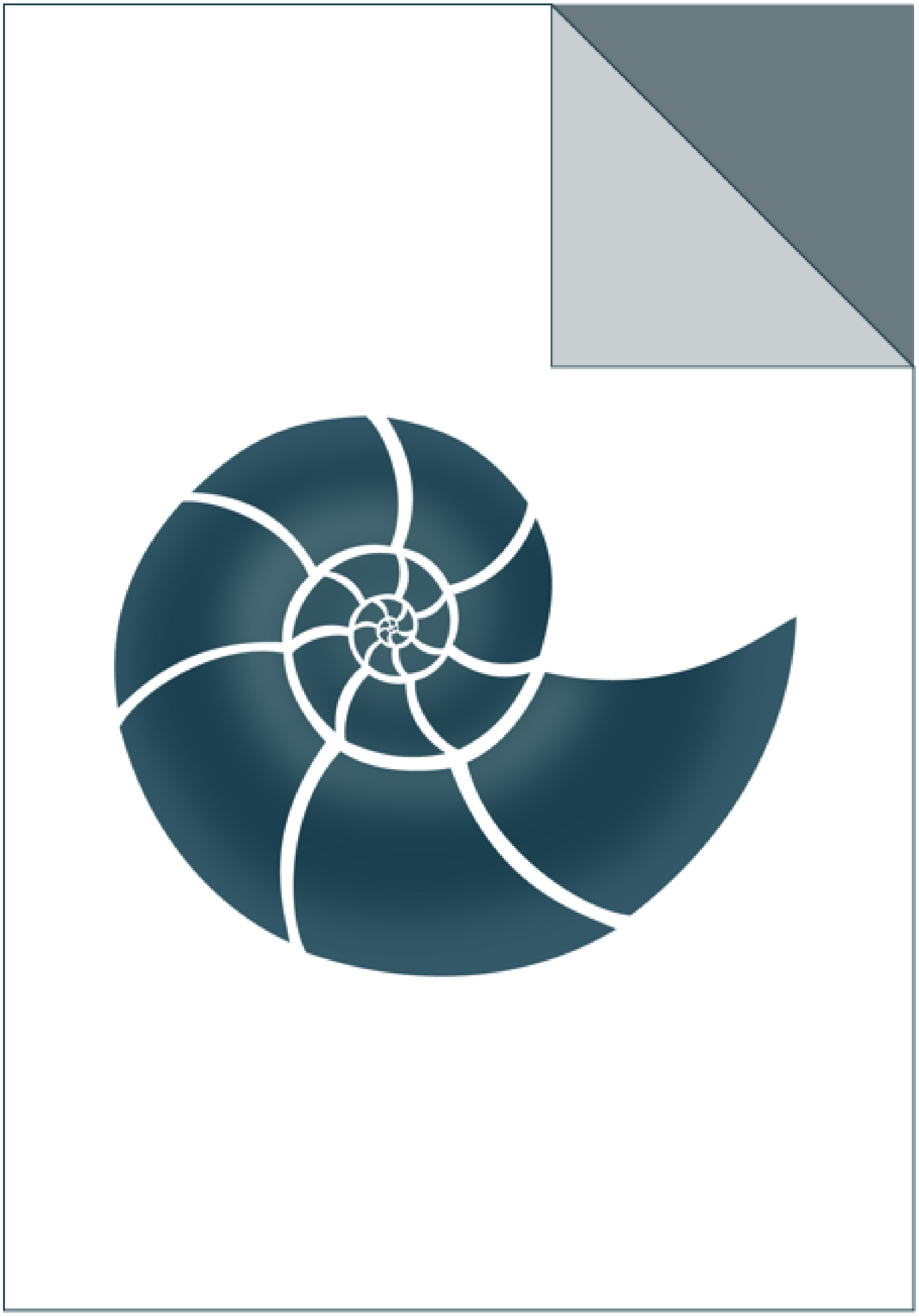
USAGE: ./ex_seq_io 2gb1.seq

Keywords:

- no_keywords

Categories:

- no_categories



5.2.83 ex_set_dihedral

Sets a particular values for Phi, Psi and Omega angles at a certain residue in a protein.

USAGE: 2gb1.pdb 18 -80.4 90.4 180.0

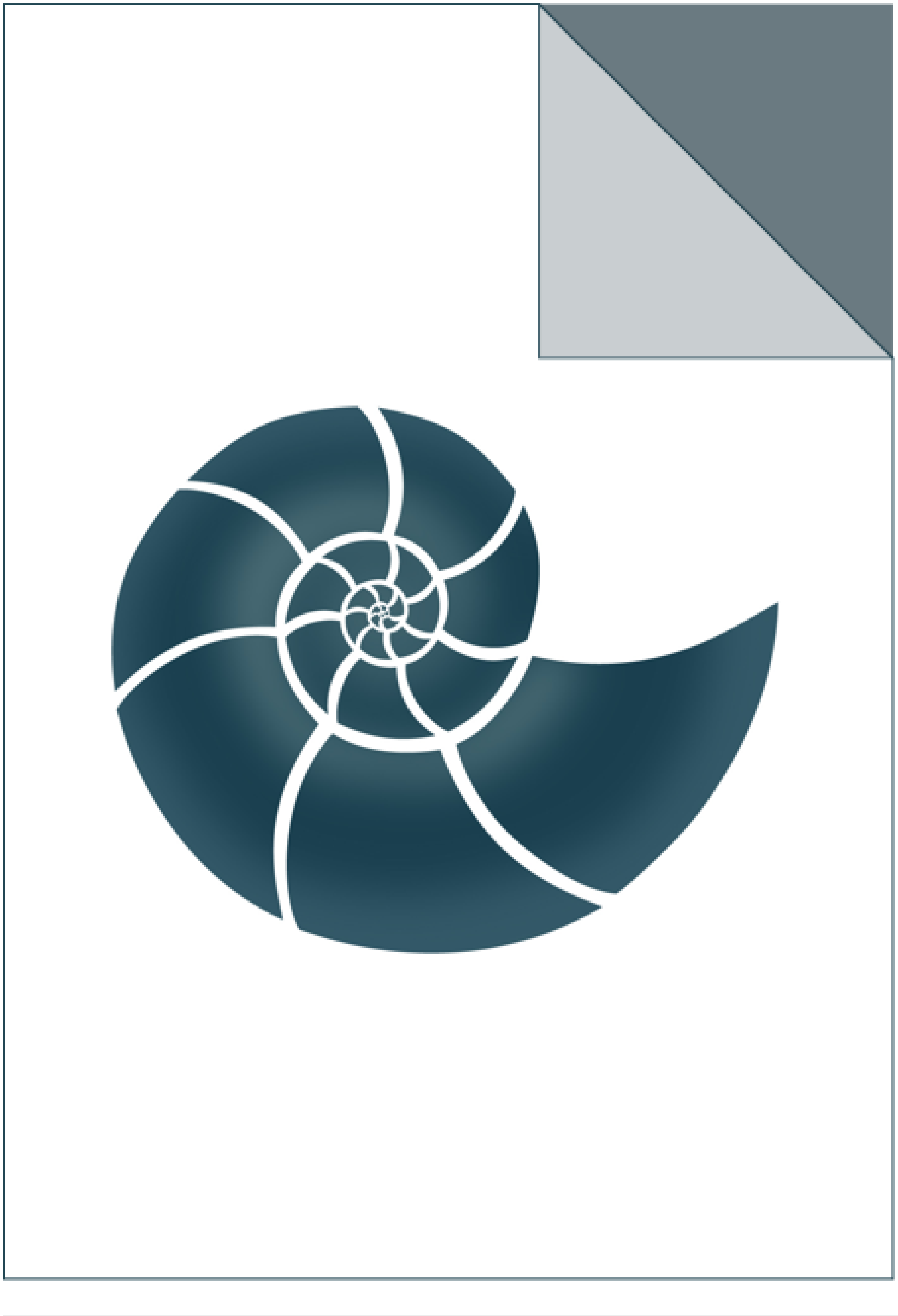
where 2gb1.pdb is the protein structure to be modified, 18 is the residue ID and the three following real values are Phi, Psi and omega dihedrals. The results is printed in PDB forma

Keywords:

- no_keywords

Categories:

- no_categories



5.2.84 ex_shared_pointers

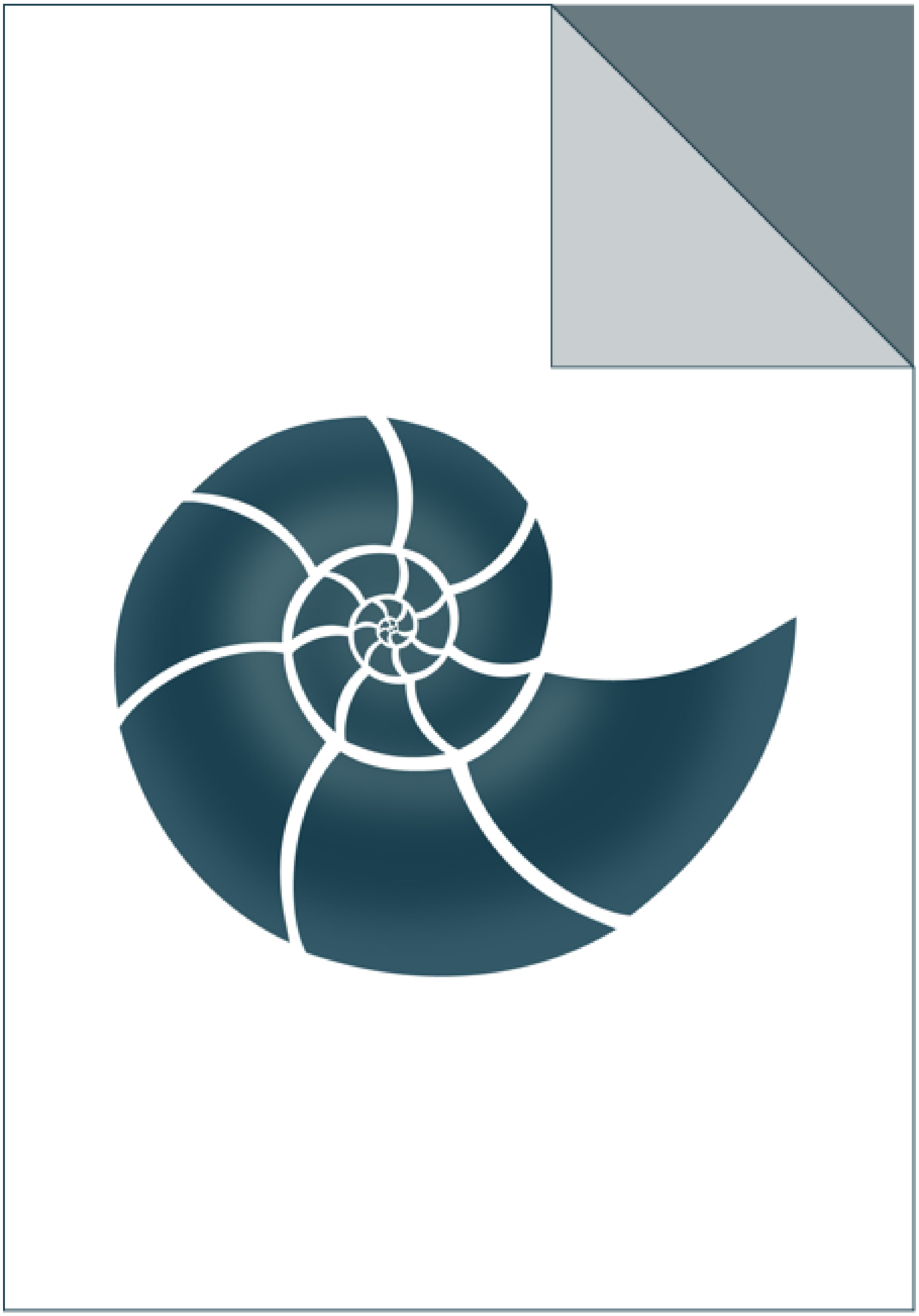
A very basic example showing how to use shared pointers (from standard C++ 11 library) when programming in BioShell.

Keywords:

- no_keywords

Categories:

- no_categories



5.2.85 ex_simpson_integration

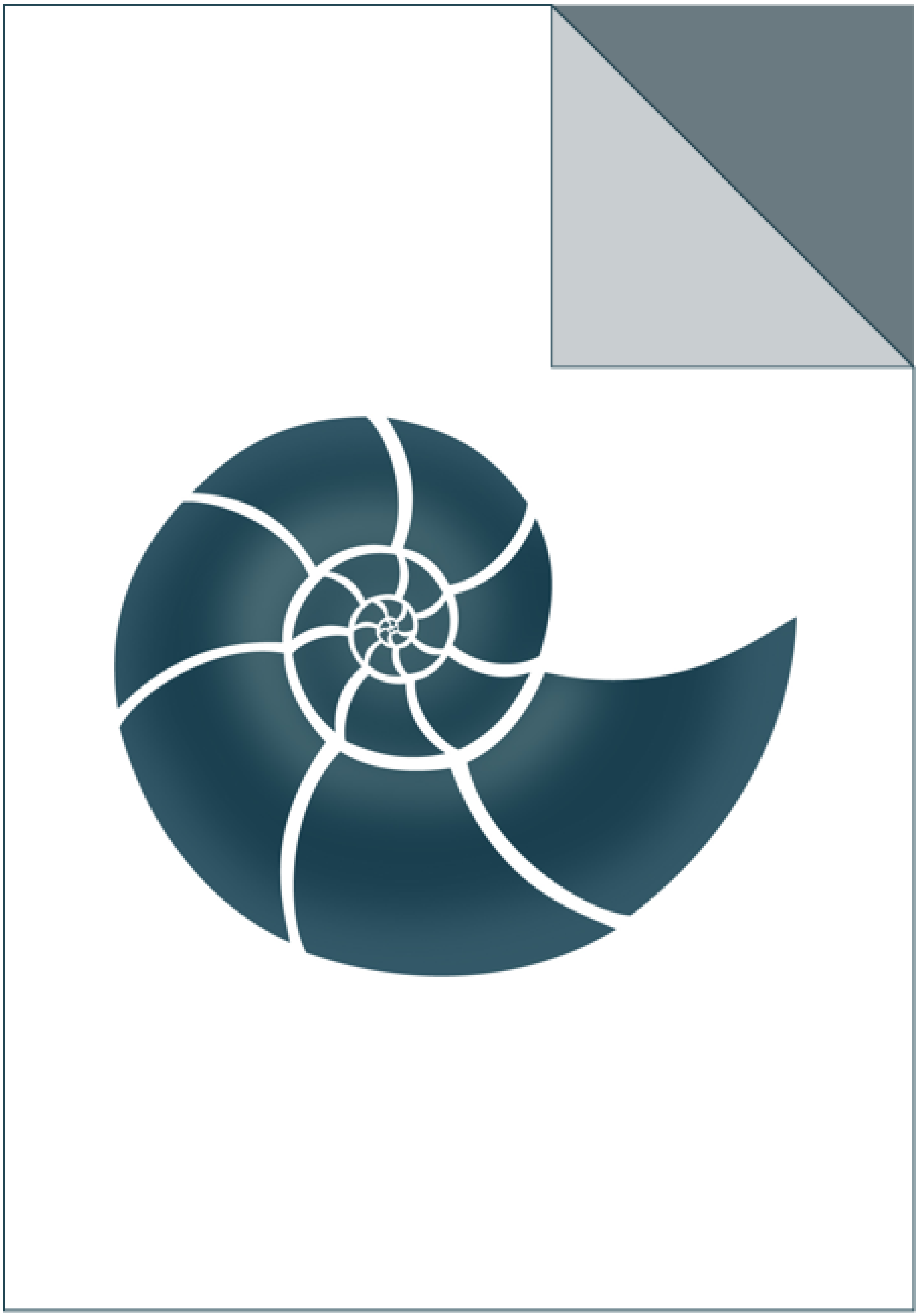
Example for numerical integration with Simpson method

Keywords:

- no_keywords

Categories:

- no_categories



5.2.86 ex_split_fasta

ex_split_fasta reads a FASTA file and writes every sequence from it in a separate file

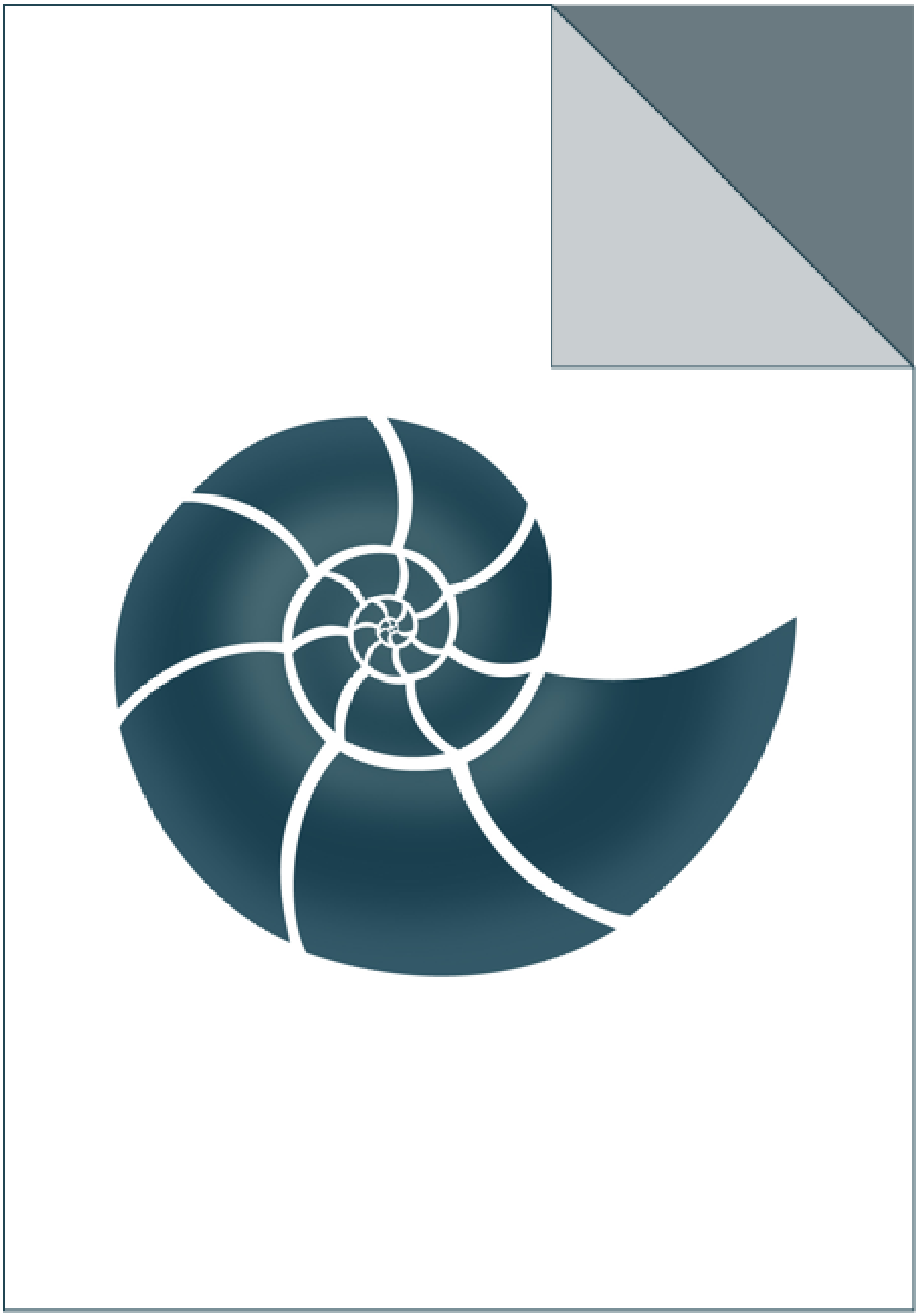
USAGE: ./ex_split_fasta 5edw.fasta

Keywords:

- no_keywords

Categories:

- no_categories



5.2.87 ex_structure_iterators

ex_structure_iterators shows how to iterate through structural components

USAGE: ex_structure_iterators 1dt7.pdb

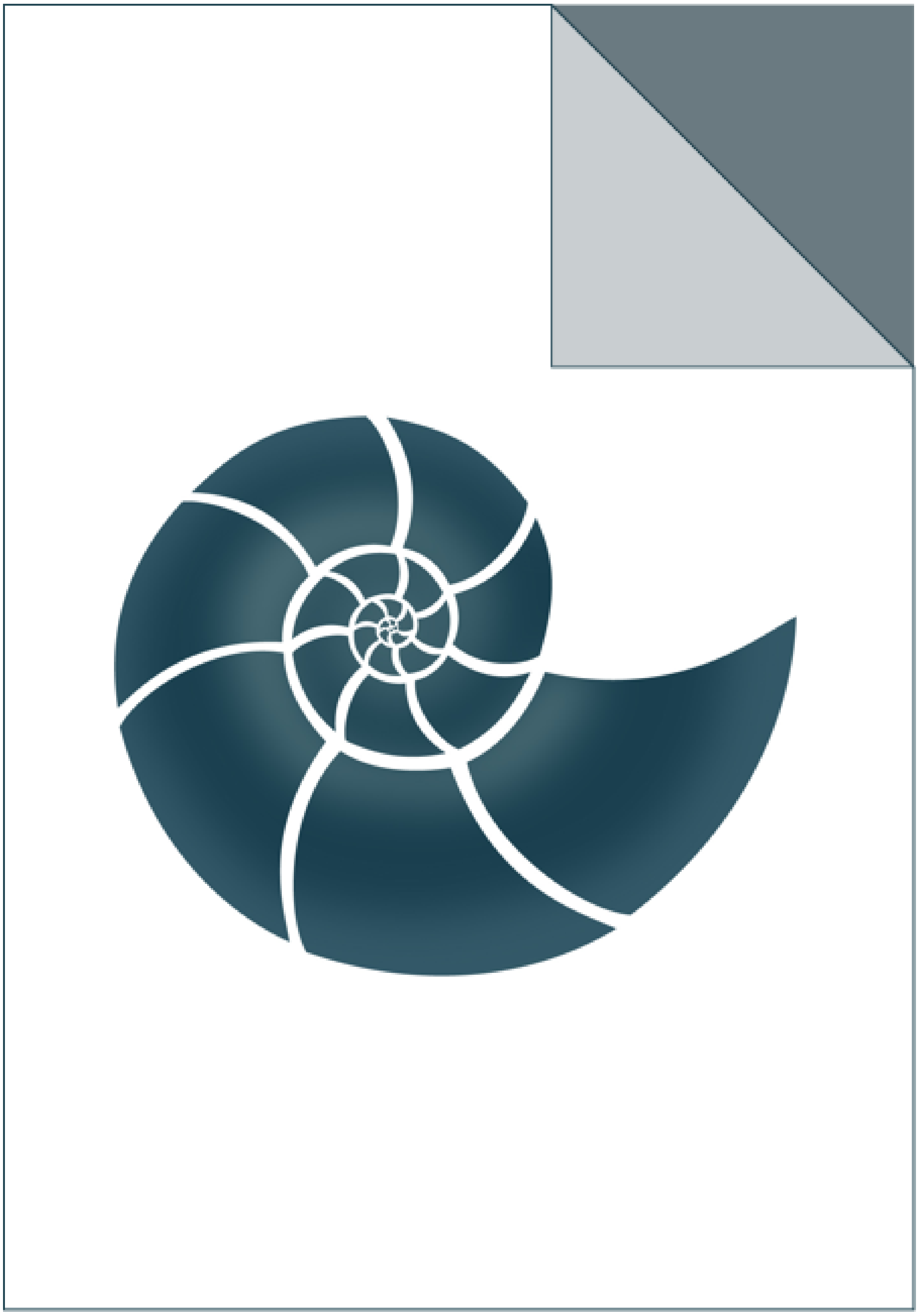
where 1dt7.pdb id an input file (PDB format)

Keywords:

- no_keywords

Categories:

- no_categories



5.2.88 ex_test_gzip

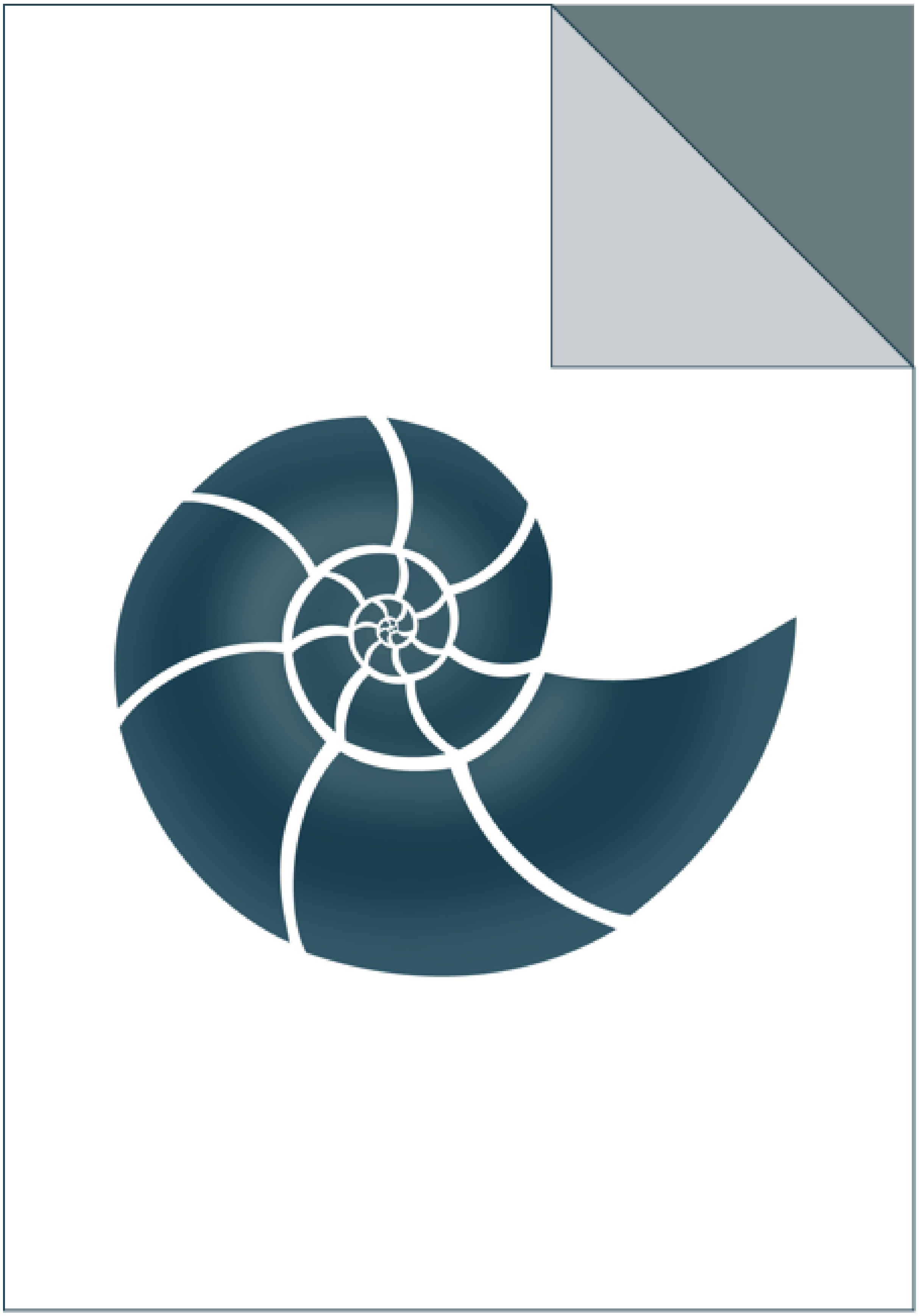
Simple test to gzip and un-gzip a string data

Keywords:

- no_keywords

Categories:

- no_categories



5.2.89 ex_uniquify

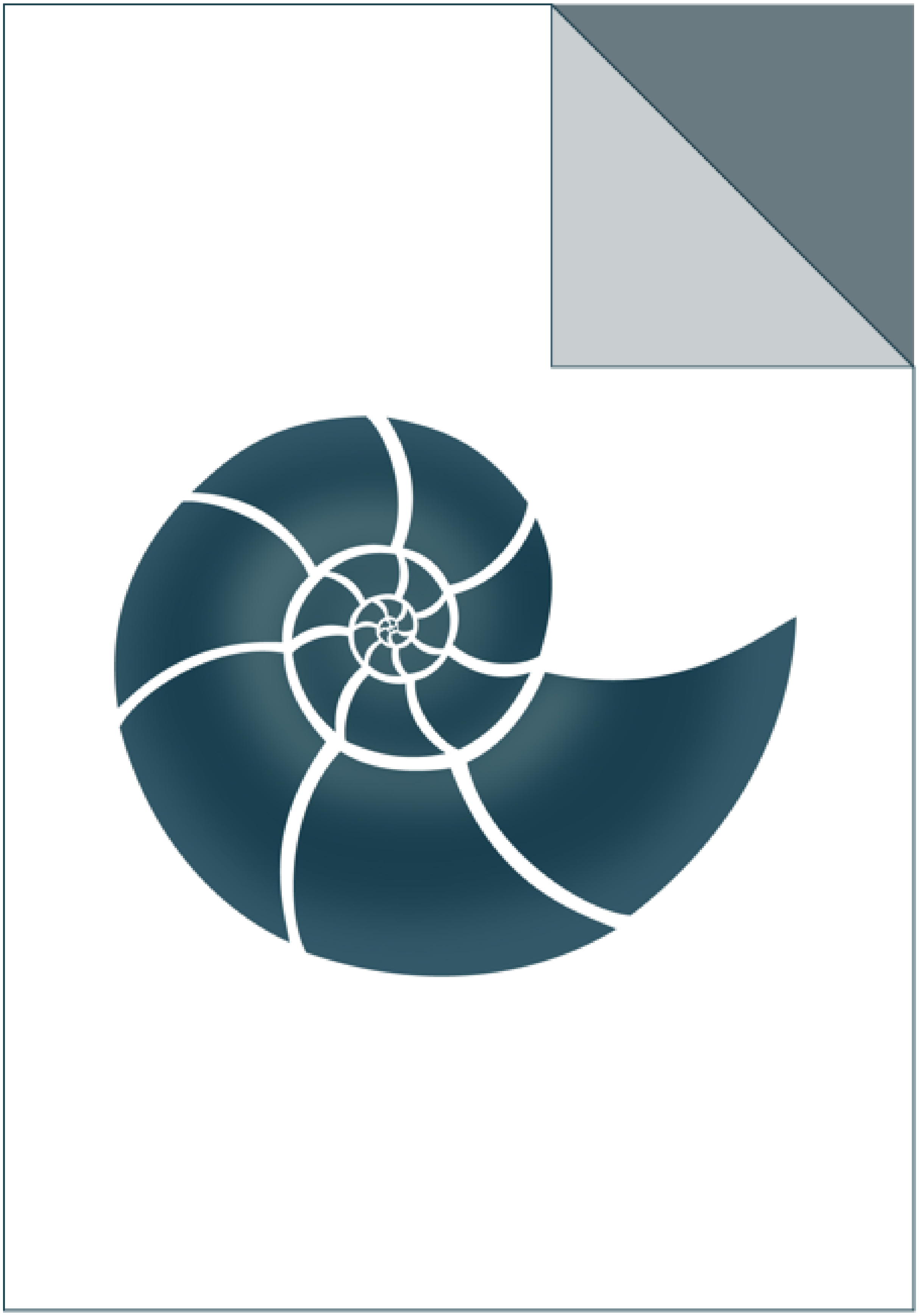
Tests uniquify() method which removes redundant objects from a container.

Keywords:

- no_keywords

Categories:

- no_categories



5.2.90 ex_web_client

Simple test for web_client methods

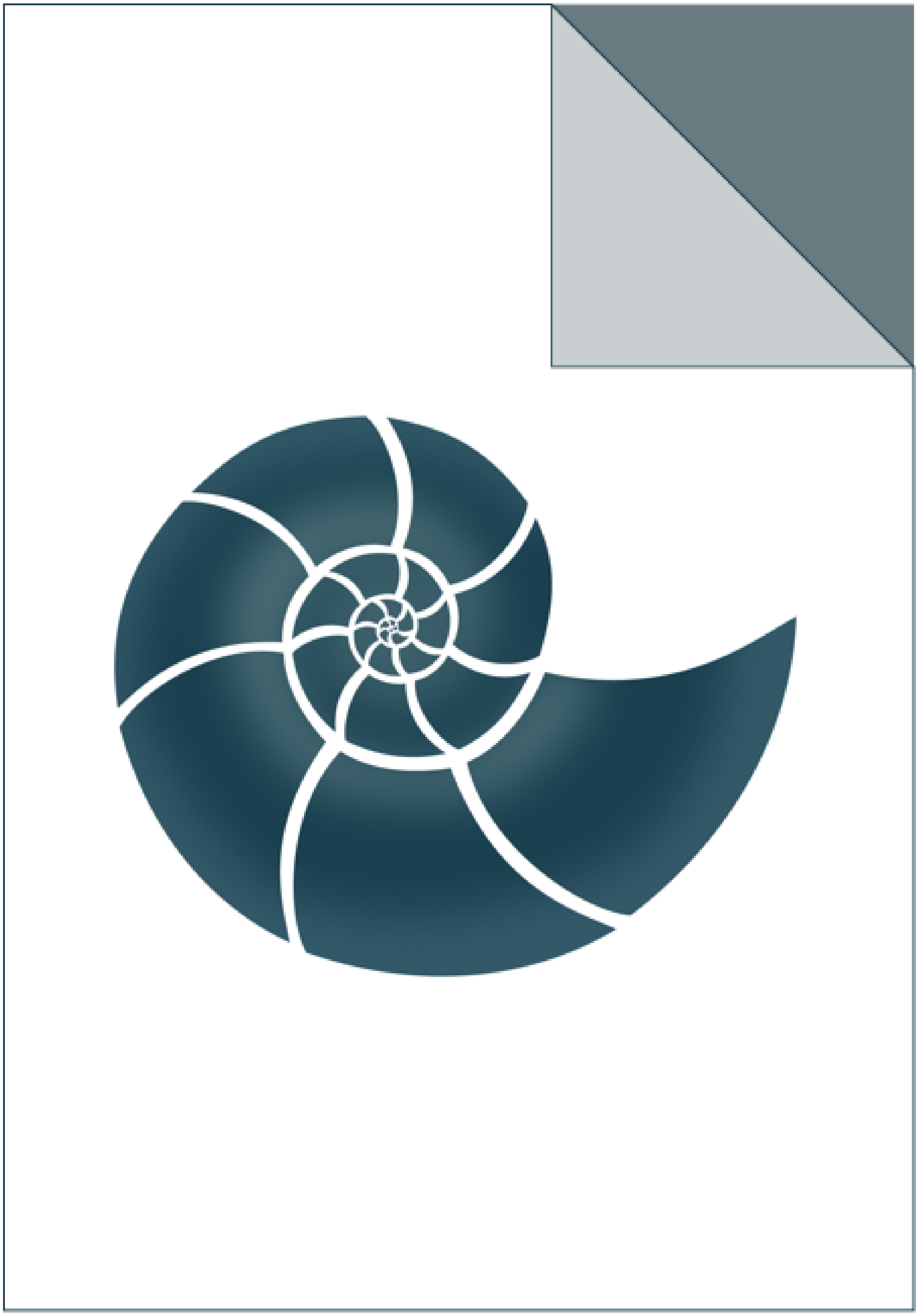
USAGE: ex_web_client [address]

Keywords:

- no_keywords

Categories:

- no_categories



5.3 *ww_** programs

These group contain test which are displayed in WWW browser.

5.3.1 ww_evaluate_chi

Calculates Chi angles for every model found in the input protein structure

and displays as a 2D scatterplot on a web page USAGE: ww_evaluate_chi [ww_evaluate_chi.htm [8002]]

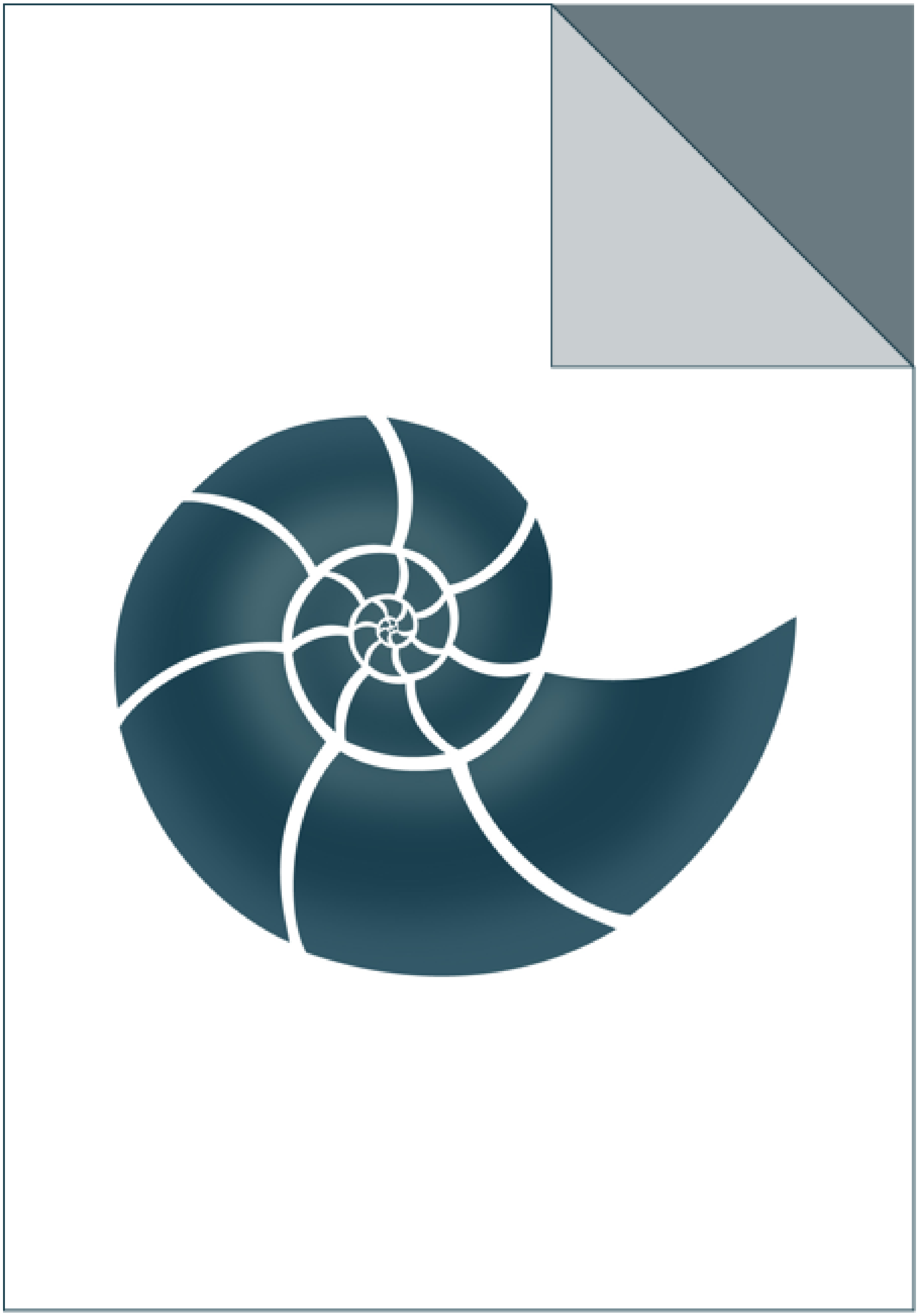
where the first optional argument is the HTML page file and 8002 is the port number (ww_evaluate_chi.htm and 8002 by default, respectively

Keywords:

- no_keywords

Categories:

- no_categories



5.3.2 ww_evaluate_contact_map

Calculates contact map for a given protein structure

USAGE: ww_evaluate_contact_map [ww_evaluate_contact_map.htm [8003 [pdb_local_path]]]

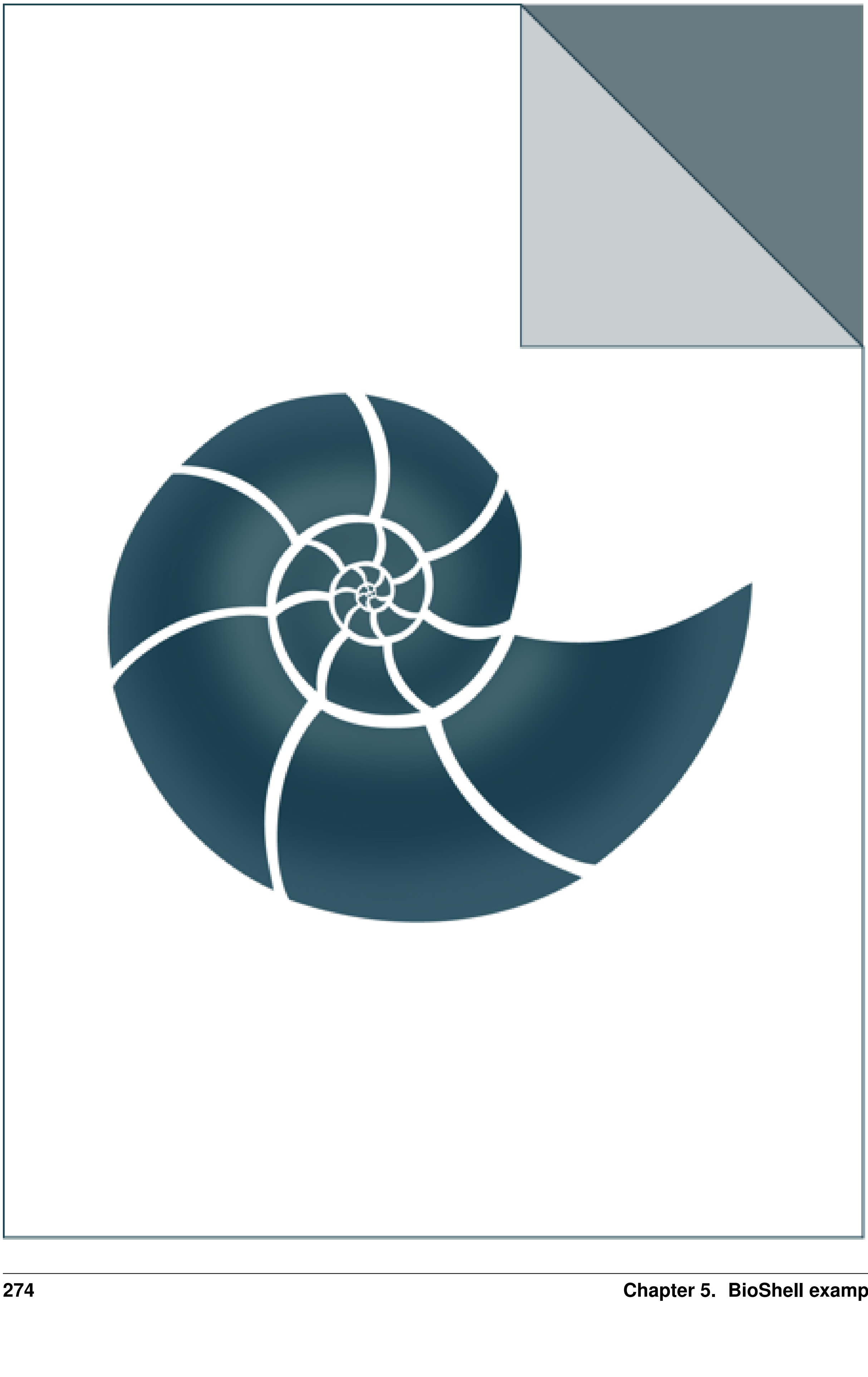
where the first optional argument is the HTML page file and 8003 is the port number (ww_evaluate_contact_map.htm and 8003 by default, respectively

Keywords:

- no_keywords

Categories:

- no_categories



5.3.3 ww_evaluate_phi_psi

Calculates Phi,Psi angles (Ramachandran map) for every model found in the input protein structure

and displays a Ramachandran plot as a web page USAGE: ww_evaluate_phi_psi [ww_evaluate_phi_psi.htm [8001 [pdb_local_path]]]

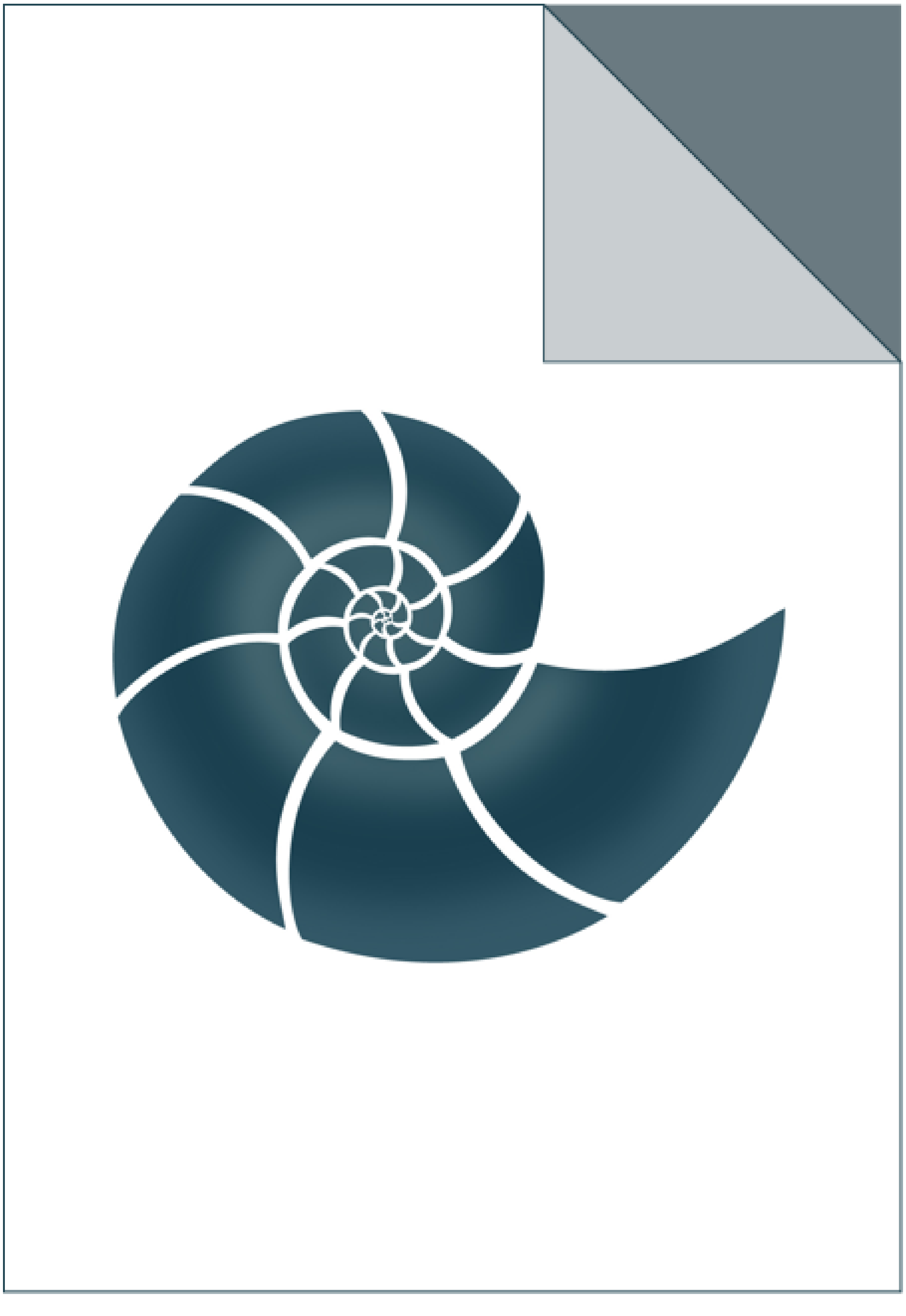
where the first optional argument is the HTML page file and 8001 is the port number (ww_evaluate_phi_psi.htm and 8001 by default, respectively

Keywords:

- no_keywords

Categories:

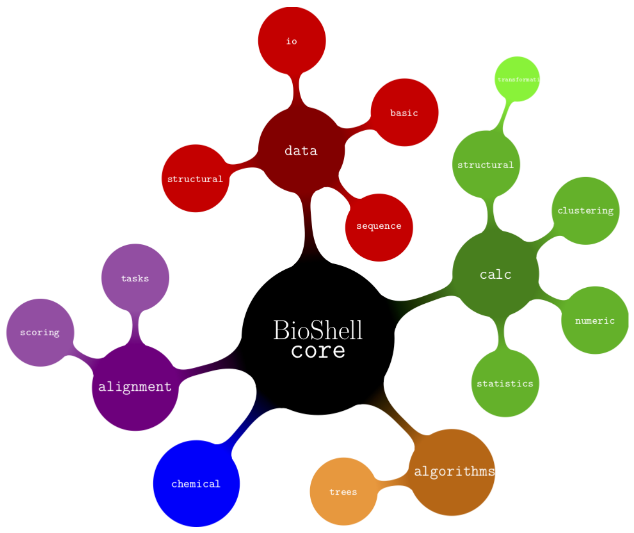
- no_categories



CHAPTER 6

BioShell C++ library

BioShell is a versatile C++11 library for structural bioinformatics. Its struture has been shown in the figure below:



6.1 Reading and processing PDB files

Reading PDB files into a BioShell program is divided into two steps:

- loading a text file into memory, and
- parsing its content and creating Structure object(s)

6.1.1 Loading a PDB file

You have to create a reader object to read a PDB file. In the simplest case this looks as below:

```
core::data::io::Pdb reader("infile.pdb");
```

This reader will skip water molecules and hydrogen atoms. You can control which PDB line will be omitted during reading by providing a **PdbLineFilter** instance to the constructor, e.g.

```
core::data::io::Pdb reader("infile.pdb",
  core::data::io::all_true(core::data::io::is_not_water,
  core::data::io::is_not_alternative));
```

`PdbLineFilter` objects can dramaticly limit the number of PDB lines to be parsed and thus shorten the time spent of PDB file loading.

6.1.2 Creating Structure object

Once a file is loaded, you can create a **Structure** object from one of its models:

```
core::data::structural::Structure_SP model = reader.create_structure(0);
```

The very first model is indexed by 0. Every time `create_structure()` method is called, a new `Structure` object is created, which includes necessary memory allocation. Creating new atom objects is in fact the slowest part of this call. Sometimes it is possible to *recycle* old structure filling it with new coordinates rather than just creating a new one from scratch. This can be done as in the `ap_contact_map` program; the relevant fragment is shown below:

Coordinates of a new structure must fit into the existing stucture i.e. the new structure must be composed of the same number of chains, residues and atom as the old one. In practice this is most useful when a multi-model PDB file must be loaded, as in this example:

- in the **line 1** a PDB file is loaded with a filter instance defined someehere before
- in the **line 3** a `Structure` object is creaded based on the first model defined in the file
- in the **line 4** a `ContactMap` object is creaded and the first structure is loaded id
- finally, in **lines 5-8** a loop iterates over all the remaining models; in **line 6** coordinates of each model are loaded into the existing structure (the one created in **line 3**)

Residue, PdbAtom and Chain objects are created only once, when the structure at index 0 is loaded. After that the loop only substitutes. coordinates of this structure

CHAPTER 7

BioShell Python library

BioShell 3.0 comes also with Python bindings i.e. BioShell classes can be also used as Python modules. Let's consider the following C++ program that reads a PDB file and writes a FASTA sequence for every chain:

The same program written in Pyton looks much simpler. It calls nearly the same BioShell C++ objects as the one above, but due to simplicity of Python, the script is a bit shorter:

7.1 Reading and writing PDB files

7.1.1 Reading PDB files

Reading PDB data is a two-stage process: first you crete a reader that loads PDB content into memory; then the content is parsed according to user's requests.

7.1.2 Writing PDB files

PdbAtom class provides `create_pdb_line()` method.

This page provides documentation for BioShell package. Laboratory protocols and documentation to Rosetta is provided by labnotes website.

CHAPTER 8

SURPASS model

SURPASS model Single United Residue per Pre-Averaged Secondary Structure fragment is a coarse-grained low resolution model for protein simulations.

- **see** doc_surpass_representation
- **read about**: doc_surpass_force_field
- **necessary and optional** doc_biosimulations_surpass_input
- **resutling** doc_biosimulations_surpass_output
- `surpass_annealing` command line program doc_biosimulations_surpass_annealing

CHAPTER 9

Indices and tables

- genindex
- search

Index