Recipe 3.9. Rendering the alignment between an experimental spectrum and a theoretical spectrum generated from a peptide

Table of contents

1. Problem

You want to render the graph chart of the alignment between a peak list against theoretical peptide fragments.

2. Solution

The object `MSPeptideMatchRenderer` is making all the steps for you, from the generation of the theoretical spectrum to the rendering of the graph:

```
import org.expasy.jpl.msmatch.MSPeptideMatchRenderer;

// the experimental peak list that we want to match against the theoretical
fragments
PeakList expPeakList = ...

// the peptide precursor that will be fragmented by the renderer
JPLIonizedPeptide peptide =
        JPLIonizedPeptide.newInstance(new Peptide.Builder(
                "EQVQSC({57})GPPPELLNGNVK").build(), 2);

// The builder defines the settings for:
// 1. the fragmentation of precursor (fragmenter, condition)
// 2. the matching of spectra (tolerance)
// 3. the rendering of the chart image (bgColor, dimension)
MSPeptideMatchRenderer renderer =
        new MSPeptideMatchRenderer.Builder().fragmenter(
                new PeptideFragmenter.Builder(EnumSet.of(
                        FragmentationType.AX, FragmentationType.BY,
                    FragmentationType.CZ)).enableLoss().build())
                    .tolerance(0.1).bgColor(Color.gray).dimension(
                     new Dimension(500, 300)).build();

// setting the data
renderer.setData(peptide, expPeakList);

// rendering the chart
try {
        renderer.exportChart("/tmp/alignment");
} catch (ImageRenderingException e) {
        // no match -> nothing to render
        e.printStackTrace();
}
```

As the whole processing of fragmentation + matching is delegated internally to `MSPeptideMatchFactory`, you can also give it to the builder and define only the rendering parameters:

```
import org.expasy.jpl.msmatch.MSPeptideMatchFactory;
```

Recipe 3.9. Rendering the alignment between an experimental spectrum and a theoretical spectrum generated from a peptide

```
import org.expasy.jpl.msmatch.MSPeptideMatchRenderer;

// the experimental peak list that we want to match against the theoretical
fragments
PeakList expPeakList = ...

// the peptide precursor that will be fragmented by the renderer
JPLIonizedPeptide peptide =
        JPLIonizedPeptide.newInstance(new Peptide.Builder(
                "EQVQSC({57})GPPPELLNGNVK").build(), 2);

// The builder defines the settings for:
// 1. the fragmentation of precursor (fragmenter, condition)
// 2. the matching of spectra (tolerance)
MSPeptideMatchFactory factory =
        new MSPeptideMatchFactory.Builder().fragmenter(
                new PeptideFragmenter.Builder(EnumSet.of(
                        FragmentationType.AX, FragmentationType.BY,
                    FragmentationType.CZ)).enableLoss().build())
                    .tolerance(0.1).build();

MSPeptideMatchRenderer renderer =
        new MSPeptideMatchRenderer.Builder(factory).bgColor(Color.gray)
                .dimension(new Dimension(500, 300)).build();

// setting the data
renderer.setData(peptide, expPeakList);

// rendering the chart
try {
        // the renderer will delegate the process to the factory
(fragmentation + matching)
        // and then will render the alignment
        BufferedImage bi = renderer.render();
} catch (ImageRenderingException e) {
        // no match -> nothing to render
        e.printStackTrace();
}
```

Sometimes it is needed to edit the processed peak list before making the rendering. In this case, you have to call process() yourself:

```
MSPeptideMatchFactory factory =
        new MSPeptideMatchFactory.Builder().fragmenter(
                new PeptideFragmenter.Builder(EnumSet.of(
                        FragmentationType.AX, FragmentationType.BY,
                    FragmentationType.CZ)).enableLoss().build())
                    .tolerance(0.1).build();

// make the computation
factory.process(peptide, expPeakList);

// get the alignment spectra
PeakList apl = factory.getAnnotatedPeakList();
```

```
PeakList napl = factory.getNonAnnotatedPeakList();

// edit the spectra
.....

// new renderer
MSPeptideMatchRenderer renderer =
        new MSPeptideMatchRenderer.Builder(factory).build();

renderer.addDataSet(apl, "annot");
renderer.addDataSet(napl, "na");

// only render the modified alignment
renderer.render();
```

Note:

If no factory has been passed to the MSPeptideMatchRenderer builder, it is also possible to get the internal one with the MSPeptideMatchRenderer accessor getFactory().

3. Discussion

See also the recipe about processing the alignment of MS spectrum against peptide fragments.

4. See Also