Web-based 3D molecular viewers: an update

Before:

- Only desktop applications could make use of GPU-based hardware acceleration through OpenGL (Chimera, VMD, PyMol etc...)
- Web browsers couldn't display 3D content without plug-ins
- Development of java applets-based viewers:
  - Jmol
  - OpenAstex Viewer

Recently:

- Java install base is shrinking due to heavily publicized java security failures
- User is annoyed with multiple security prompt
- Java applets are no longer supported in Chrome
- Development of JavaScript-based viewers:
  - Jmol → java2script (java to javascript converter) → Jsmol:
    - Same as Jmol but very slow
    - Old fashioned

Now:

- Modern browsers can use hardware-accelerated graphics through the WebGL standard :
  - Firefox since version 5 (2010)
  - Chrome since version 9 (2011)
  - Safari since version 5.1 (2011)
  - Opera since version 12 (2012)
  - Internet Explorer since version 11 (2013)

- GLMol is the first Web-based 3D molecule viewer to make use of WebGL
  - Relies on three.js library
  - Still relatively slow

- New WebGL viewers start to appear since 2013 :
  - PV
  - NGL
  - 3DMol.js
  - Molmil
  - Jolecule

Usage:

	Citations	Notable sites using it
NGL	32	RCSB(PDB)
3Dmol.js	27	CATH
Molmil	4	PDBj
PV	N/A	RCSB(PDB) SWISS-MODEL
Jolecule	N/A	?

Features:

	PV	NGL	3DMol.js	MolMil
File supports	pdb, sdf	pdb, sdf, mol2, pqr, gro, cif, mmtf	pdb, sdf, mol2, xyz	pdb, cif, gro, trr, xtc, cor, mpbf, ccp4,mdl, mol2
Molecular surface		✓	✓	✓
Sphere, stick, line, trace, cartoon etc...	✓	✓	✓	✓
Clickable interactivity	✓	✓	✓	✓
Geometric shapes	✓	✓	✓	
Hosted viewer		✓	✓	✓

Performance comparison:

	JMol	JSMol	GLMol	PV	NGL	3Dmol.js
Construct	0.136s	0.874s	6.372s	2.146	1.095s	0.776s
Rotate	0.053s	0.207s	0.673s	0.001s	0.002s	0.002s

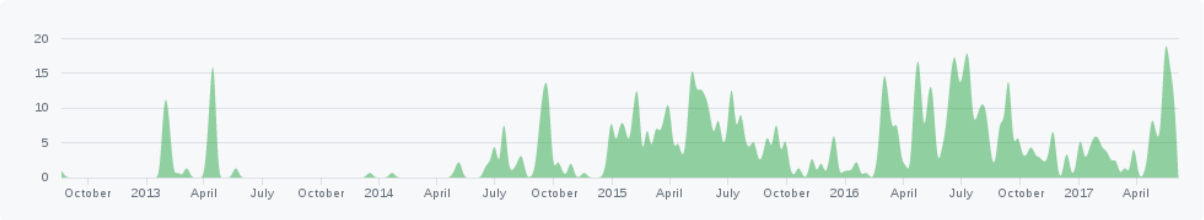
Time to load 3M8L (capsid protein) with 12,375 atoms and then rotate it.

Conclusion : 3Dmol.js may be a bit faster at building large molecules. But once loaded, PV, NGL and 3Dmol.js performances are rather similar.

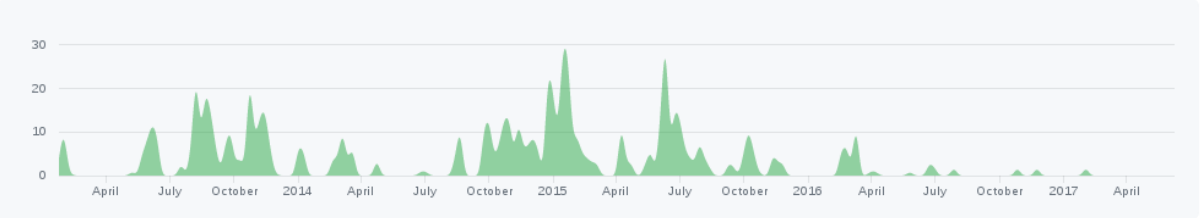
Development activity:

NGL

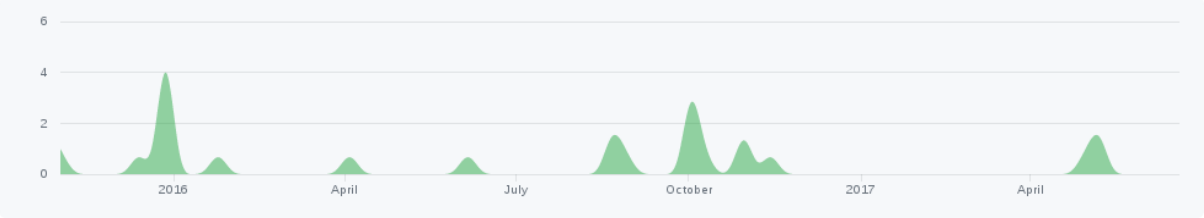
3Dmol.js



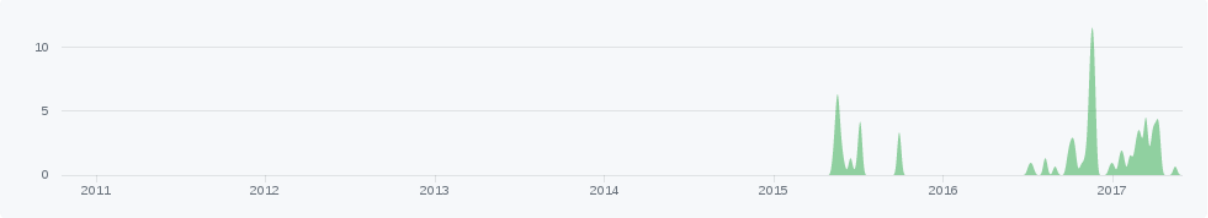
PV

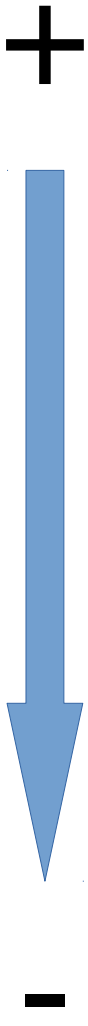


Molmil



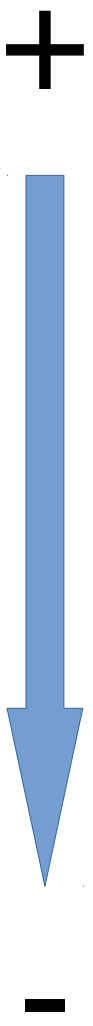
Jolecule





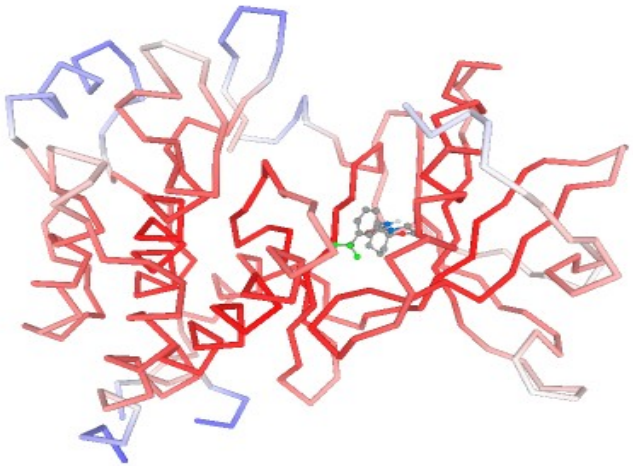
API usage:

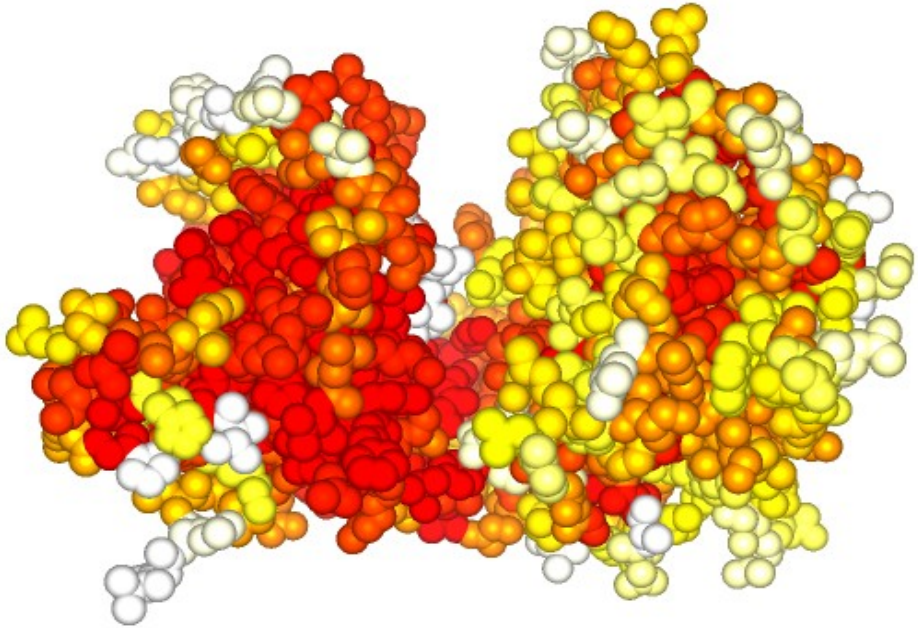
	deployment speed	documentation	examples
NGL	++	++	+++
3Dmol.js	+++	++	+
PV	++	+++	-
Molmil	?	+	-



PV:

- https://github.com/biasmv/pv
- Developed by Marco Biasini (+ 12 other contributors)
- MIT license
- Very light (156 K)
- Beautiful renderings
- Oriented towards proteins only
- Embedded viewer only (no hosted viewer)
- No longer developped (last commit : Feb 23rd 2017)
- Lacked some features at the beginning like multi-model pdb files management, coloration by b-factor, etc.
- Still lacks surfaces rendering





PV:

Installation:

```
git clone https://github.com/biasmv/pv
```

Invocation:

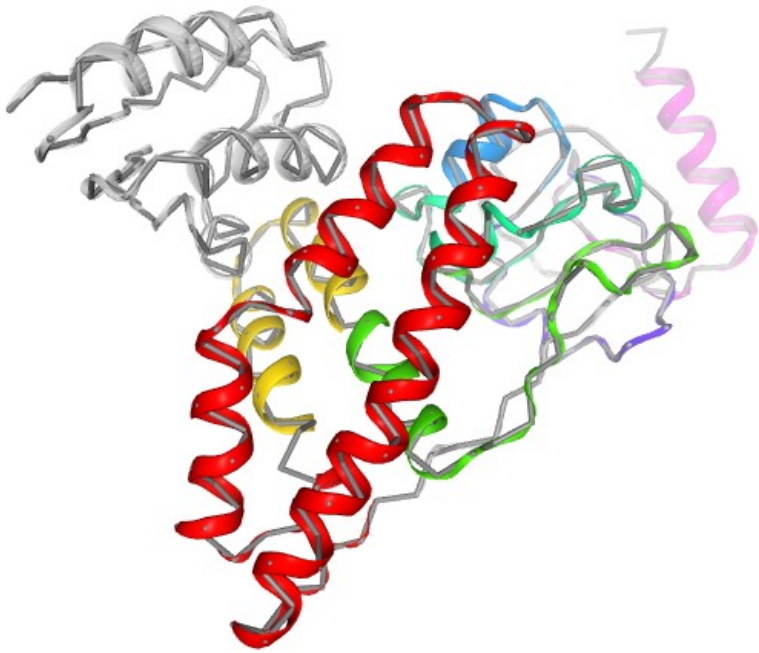
```
<body>
  <div id="viewport"></div>
</body>

<script src='jquery.js'></script>
<script src='pv.min.js'></script>

<script>
  var pv = pv.Viewer(document.getElementById("viewport"), {
    width: 658,
    height: 616,
    antialias: true,
    quality: 'high',
    outline: false
  });
  var structure;
  $(document).ready(function() {
    $.ajax({ url : file, success : function(data) {
      structure = io.pdb(data);
      pv.cartoon("cartoon", structure, { color: color.ssSuccession() });
    }}) ;
  });
</script>
```

NGL:

- https://github.com/arose/ngl
- Developped by AS. Rose and PW. Hildebrand (Charité–Universitätsmedizin Berlin, Germany)
- MIT license
- Can read both proteins and RNA/DNA molecules
- Very well documented, gallery with lots of examples
- Very complete
- Embedded viewer as well as hosted viewer :

http://proteinformatics.charite.de/ngl-tools/ngl/html/ngl.htr

- Maybe not as pretty as PV (more aliasing)



Rose, A. S., & Hildebrand, P. W. (2015). NGL Viewer: a web application for molecular visualization. Nucleic acids research, 43(W1), W576-W579.

NGL:

Installation:

```
git clone https://github.com/arose/ngl
```

Invocation:

```
<body>
  <div id="viewport"></div>
</body>

<script src='jquery.js'></script>
<script src='ngl.js'></script>

<script type="text/javascript">
  var stage = new NGL.Stage("viewport");
  stage.viewer.setBackground("white");
  $(document).ready(function() {
    stage.loadFile(data).then( function (o) {
      o.addRepresentation("cartoon", { color: "atomindex" });
      o.autoView();
    });
  });
</script>
```

3Dmol.js:

- https://github.com/3dmol/3Dmol.js
- Developed by N. Rego and D. Koes (University of Pittsburgh, USA)
- BSD licence
- Looks very promising (not tested yet!)
- Hosted viewer at http://3dmol.csb.pitt.edu/viewer.html
- Can be used in 3 ways :

JavaScript API

```
$3Dmol.download('pdb:3M8L', viewer);
viewer.setStyle({chain:'A'}, {sphere:{} });
viewer.setStyle({chain:'B'},
            {cartoon:{color: 'spectrum'} });
viewer.setStyle({chain:'C'}, {stick:{} });
viewer.render();
```

HTML

```
<div style="height: 600px; width: 600px;"
 class='viewer_3Dmoljs' data-pdb='3M8L'
 data-backgroundcolor='0xffffff'
 data-select1='chain:A'
 data-style1='sphere'
 data-select2='chain:B'
 data-style2='cartoon:color=spectrum'
 data-select3='chain:C'
 data-style3='stick'></div>
```

Hosted viewer

```
http://3dmol.csb.pitt.edu/viewer.html?
 pdb=3M8L&select=chain:A&style=sphere&
 select=chain:B&style=cartoon:color~spectrum&
 select=chain:C&style=stick
```

Rego, N., & Koes, D. (2014). 3Dmol. js: molecular visualization with WebGL. Bioinformatics, 31(8), 1322-1324.

3Dmol.js:

Installation:

```
Can be invoked directly from http://3Dmol.csb.pitt.edu/build/3Dmol-min.js

Or downloaded:

git clone https://github.com/dkoes/3Dmol.js.git
```

Invocation:

```
<body>
  <div id="viewport"></div>
</body>

<script src="http://3Dmol.csb.pitt.edu/build/3Dmol-min.js"></script>

<script type="text/javascript">
  var viewer = $3Dmol.createViewer($("#div"));
  $3Dmol.download("pdb:1MO8",viewer,{multimodel:true, frames:true},function(){
     viewer.setStyle({}, {cartoon:{color:"spectrum"}});
     viewer.render();
     viewer.addSurface($3Dmol.SurfaceType.VDW, {
       opacity:0.85,
       voldata: new $3Dmol.VolumeData(volumedata, "cube"),
       volscheme: new $3Dmol.Gradient.ROYGB(range[1],range[0])
     },{});
  });
</script>
```

Integration into Mobyle / Galaxy

Mobyle	Galaxy
• Type-specific visualizations (applets) • Service-specific visualizations	• Type-specific visualizations

Integration into Mobyle:

Type-specific visualizations are set up with XML files

/opt/mobyle/core/Local/Services/Viewers/pv.xml

```
  <interface type="viewer">
    <center xmlns="http://www.w3.org/1999/xhtml" style="height: 100%">
    <applet codebase="/data/services/servers/local/viewers/jmol" code="JmolApplet" archive="JmolApplet.jar">
          <param data-parametername="structfile" name="load" value="data-url"/>
          <param name="progressBar" value="true"/>
    </applet>
    </center>
  </interface>
</head>
<parameters>
  <parameter>
    <name>structfile</name>
    <prompt>3D Structure</prompt>
    <type>
      <datatype>
        <superclass>AbstractText</superclass>
        <class>_3DStructure</class>
      </datatype>
      <dataFormat>PDB</dataFormat>
      <dataFormat>mol2</dataFormat>
      <dataFormat>sdf</dataFormat>
    </type>
  </parameter>
</parameters>
```

Integration into Mobyle:

Mobyle was designed to integrate java-applet viewers only:
can be overcome with the use of iframes

/opt/mobyle/core/Local/Services/Viewers/pv.xml

```
<parameters>
  <parameter>
    <name>structfile</name>
    <prompt>3D Structure</prompt>
    <type>
      <datatype>
        <superclass>AbstractText</superclass>
        <class>_3DStructure</class>
      </datatype>
      <dataFormat>PDB</dataFormat>
    </type>
    <interface type="viewer">
      <center xmlns="http://www.w3.org/1999/xhtml" style="height: 100%">
        <iframe frameborder="0" id="pv_frame" src="/portal/js/pv/viewer.html"/>
        <param data-parametername="structfile" name="load" value="data-url"/>
      </center>
    </interface>
  </parameter>
</parameters>
```

Integration into Mobyle:

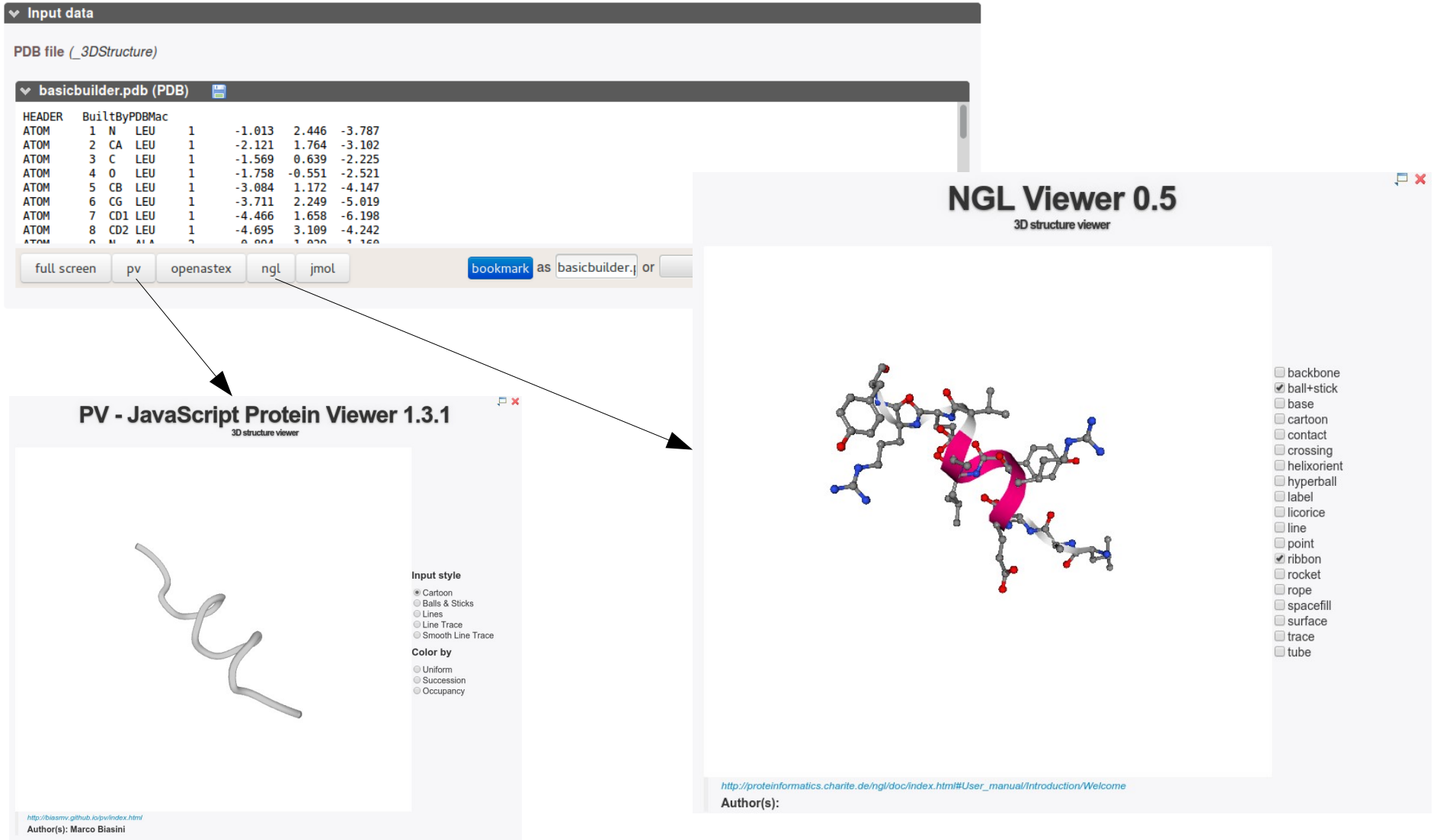
Glimpse of JavaScript code which is integrated into the iframe

/opt/mobyle/htdocs/portal/js/pv/viewer.html

```
function getURLParameter(name) {
  return decodeURIComponent((new RegExp('[?|&]' + name + '=' + '([^&;]+?)(&|#|;|$)').exec(document.referrer)||[,""])...
}

$(document).ready(function() {
  var structfile;
  var param_structfile = getURLParameter("parameter");
  if (!param_structfile) {
    structfile = $(window.frameElement).parent().find('param').prop('value');
  } else {
    structfile = param_structfile.split("|")[1];
  }
  loadModel(structfile, "protein", cartoon);
});
```

Integration into Mobyle:

Type-specific visualizations



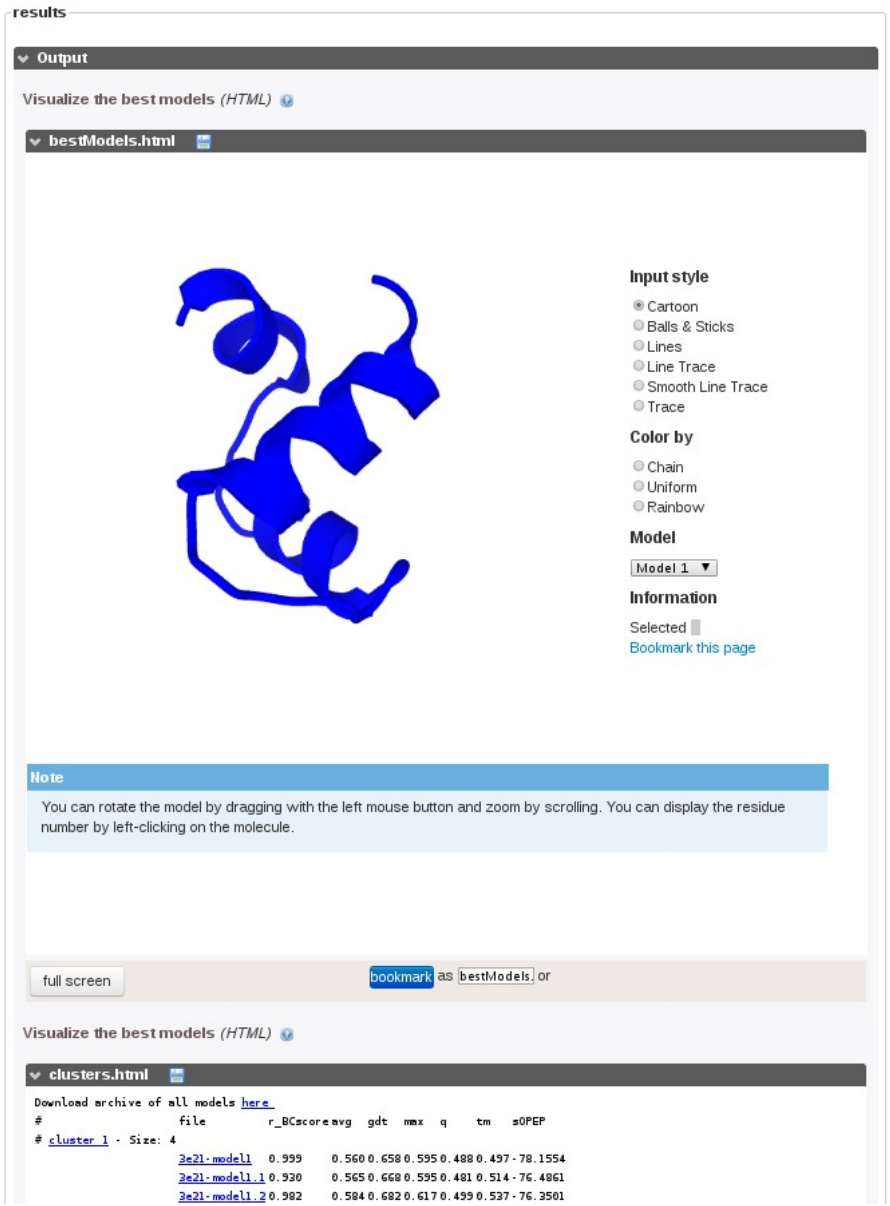
Integration into Mobyle:

Service-specific visualizations are set up with Jinja templates

/service/templates/pepfold3.vis.html

```
    $(document).ready(function() {

{% if receptor is defined %}
        loadModel('{{ wdir }}/{{ receptor }}', "receptor");
{% endif %}
        var q = queryString.parse(location.search);
        if (q.model == null) {
          q.model = $("#slider option:first").attr('value');
        }

        $("#slider select").change(function () {
          var model_file = $(this).find(":selected").attr('value');
          loadModel('{{ wdir }}/' + model_file, "protein");
          q.model = model_file;
          $('#bookmark').attr('href', location.origin + location.pathname + '?' + queryString.stringify(q));
        });
        $("#slider select").val(q.model).change();
        $('#{{ color }}').prop("checked", true);
    });
```

Integration into Mobyle:



Service-specific visualizations:

- Each service can produce its own visualization (html page)
- The result page aggregates all results
- The result page can be bookmarked

Live example:

http://mobyle.rpbs.univ-paris-diderot.fr/cgi-bin/portal.py#jobs::InterEvDock.O30995888458967

http://mobyle.rpbs.univ-paris-diderot.fr/data/jobs/MTiOpenScreen/H07673856726885/output.vis.html

Integration into Galaxy:

Type-specific viewers are set up with XML config file and mako template

/opt/galaxy/config/plugins/visualizations/pv/templates/pv.mako

```
<script type="text/javascript">
    var hdaId   = '${trans.security.encode_id( hda.id )}',
        hdaExt  = '${hda.ext}',
        ajaxUrl = "${h.url_for( controller='/datasets', action='index')}/" + hdaId + "/display?to_ext=" + hdaExt;

    var viewer = pv.Viewer(document.getElementById('gl'), {
                width: 600,
                height: 600,
                antialias: true,
                quality: 'medium',
                outline: false
              });
…
    $(document).ready(function() {
        $("#color input option:first").prop('checked', true);
        loadModel(ajaxUrl, 'main', cartoon);
    });
</script>
```

Integration into Galaxy:

Type-specific visualizations