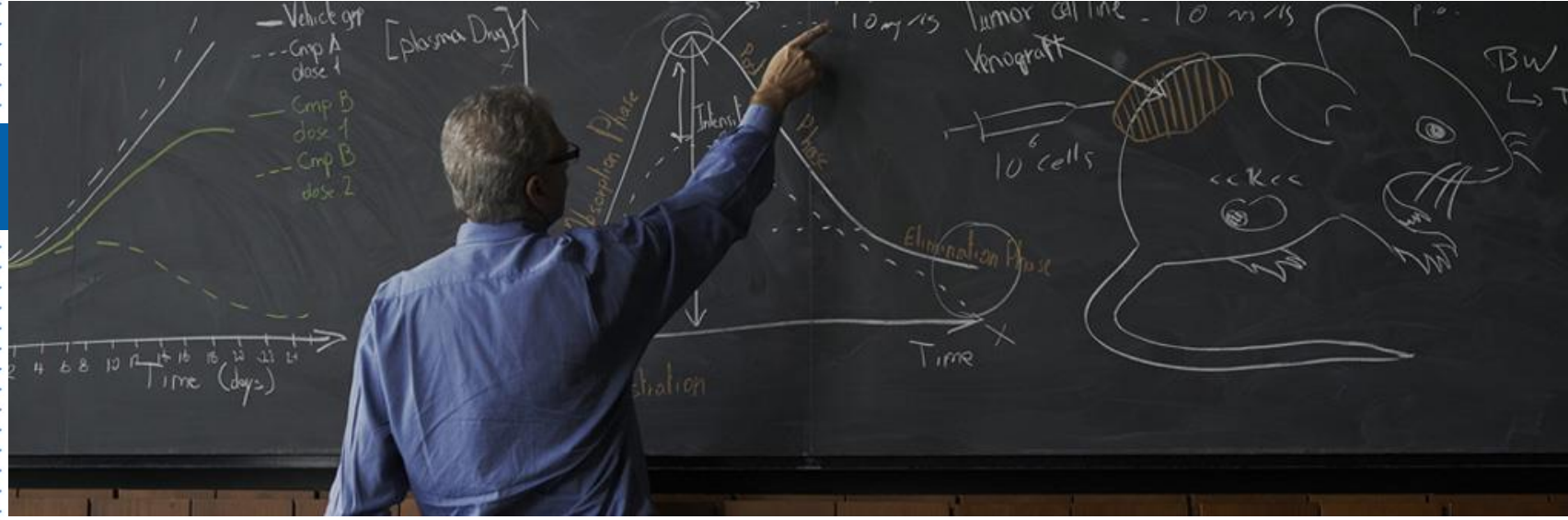
NIBR / GDC



Interactive rendering of 2D structures with RDKit

Paolo Tosco, Nico Pulver, Nik Stiefl

11th RDKit UGM

12 October 2022

NOVARTIS | Reimagining Medicine

Agenda

Where we were (2021)

Where we are (2022)

What next

Where we were (2021)

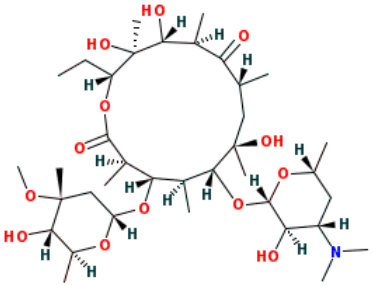
Visualize 2D molecules: from server-based...

- Several apps (Python, Java, JS, .NET) @NIBR need 2D structure visualization
- This used to be handled through a web service based on the Avalon depicter
- The client requested a 2D layout to the server (SMILES or CTAB)
- The server generated the 2D layout as a PNG image and exposed it to the client through a URL
- Several configuration options were available
  - Width, height
  - Align the layout to a scaffold (SMILES or CTAB)
  - Font, coloring, etc.

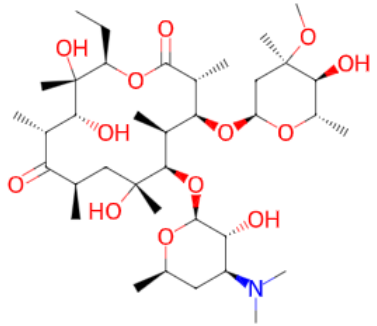
...to client-based

- Shut down the Avalon Depicter web service
- Generate all depictions in the clients (Python, JS, Java, .NET) using RDKit

```
<img width="300px" height="250px"
src="https://[...]/services/depicter/mol-
renderer/test.png?smiles=CC[C@@H]1[C@@]([C@@H]([C@H](C(=O)[C@
@H](C[C@@]([C@@H]([C@H]([C@@H]([C@H](C(=O)O1)C)O[C@H]2C[C@@](
[C@H]([C@@H](O2)C)O)(C)OC)C)O[C@H]3[C@@H]([C@H](C[C@H](O3)C)N
(C)C)O)(C)O)C)C)O)(C)O&w=300&h=250">
```

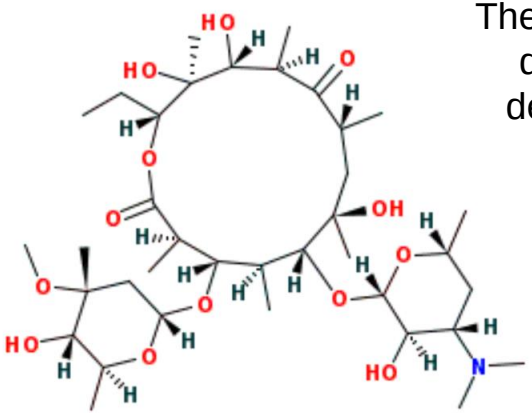


```
<div class="rdk-str-rnr-mol-container" id="rdk-str-rnr-mol-1"
data-width="300" data-height="250" data-
mol="CC[C@@H]1[C@@]([C@@H]([C@H](C(=O)[C@@H](C[C@@]([C@@H]([C
@H]([C@@H]([C@H](C(=O)O1)C)O[C@H]2C[C@@]([C@H]([C@@H](O2)C)O)
(C)OC)C)O[C@H]3[C@@H]([C@H](C[C@H](O3)C)N(C)C)O)(C)O)C)C)O)(C
)O" data-recompute2d></div>
```

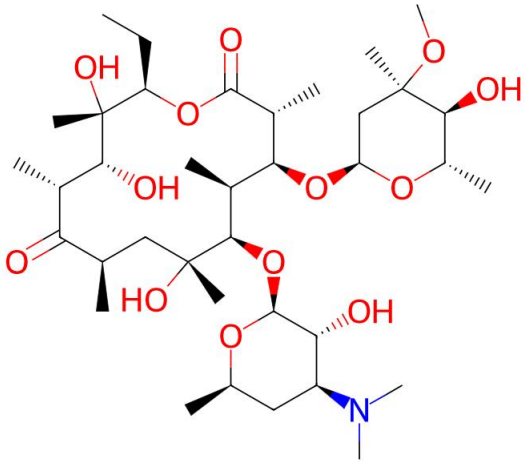


From PNG to scalable vector images

- Move from a static PNG image to a vector image (HTML5 `canvas`, SVG)



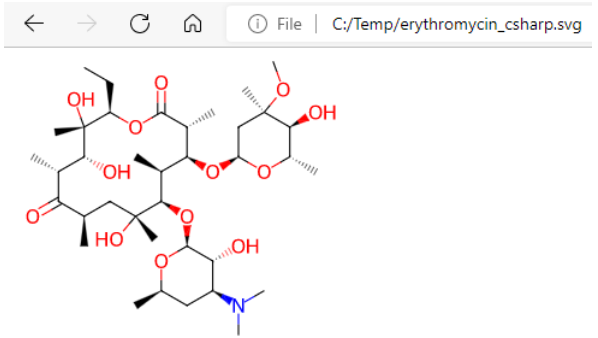
The SVG structure does not lose definition when zooming in



RDKit, RDKit everywhere

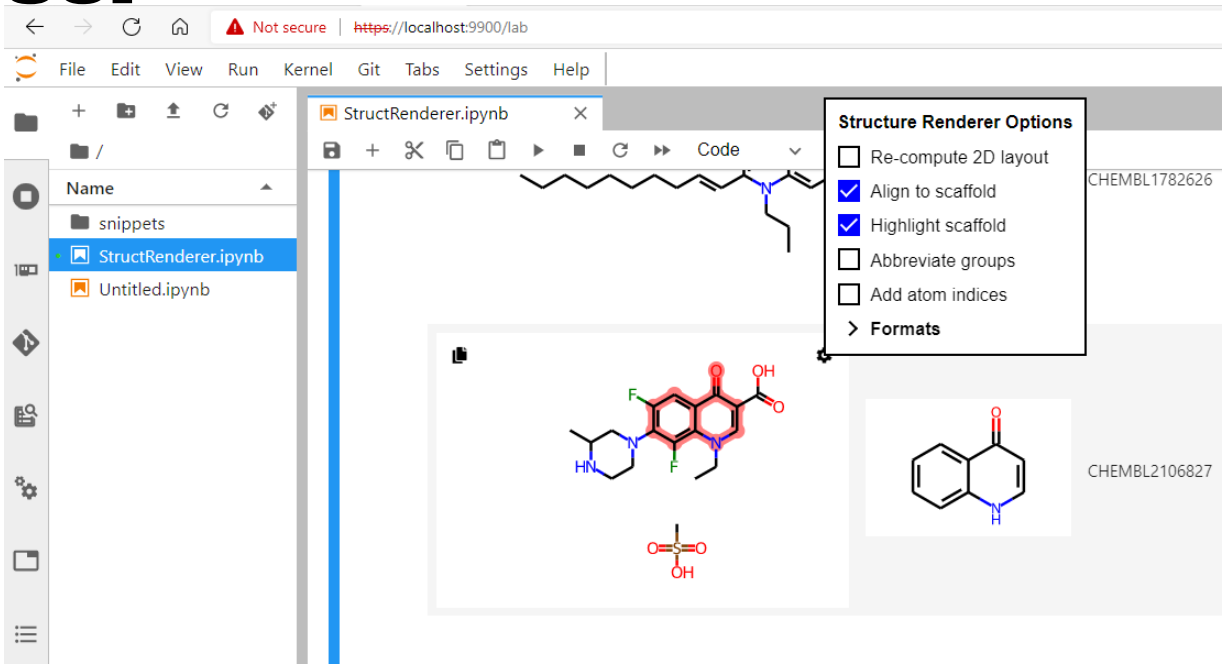
- All the base functionality is available in the RDKit C++ core
- Can be accessed from any language RDKit has wrappers for (C, Python, Java, JS, .NET)

```
ROMol erythromycin =
RWMol.MolFromSmiles("CC[C@@H]1[C@@]([C@@H]([C@H](C(=O)[C@@H](C[C@@]([C@
@H]([C@H]([C@@H]([C@H](C(=O)O1)C)O[C@H]2C[C@@]([C@H]([C@@H](O2)C)O)(C)O
C)C)O[C@H]3[C@@H]([C@H](C[C@H](O3)C)N(C)C)O)(C)O)C)C)O)(C)O");
using (MolDraw2DSVG molDraw = new MolDraw2DSVG(300, 250)) {
    RDKFuncs.setPreferCoordGen(true);
    erythromycin.compute2DCoords();
    RDKFuncs.setPreferCoordGen(false);
    erythromycin.normalizeDepiction();
    erythromycin.straightenDepiction();
    RDKFuncs.updateDrawerParamsFromJSON(molDraw, drawOptions);
    molDraw.drawMolecule(erythromycin);
    molDraw.finishDrawing();
    string svg = molDraw.getDrawingText();
    File.WriteAllText("C:/Temp/erythromycin_csharp.svg", svg);
}
```



Interactive rendering of 2D structures in the browser

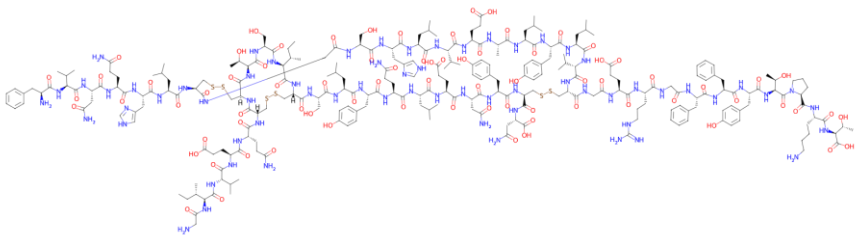
- At the 2021 RDKit UGM I presented a PoC JS library to do interactive rendering of 2D structures in the browser



Shortcomings

- My initial PoC was computing 2D layouts in the main thread
- This can be quite CPU-intensive for big hairy molecules
- The UI becomes unresponsive while the main thread is locked

```
<div class="rdk-str-rnr-mol-container structure" id="rdk-str-rnr-mol-
1" data-width="1200" data-height="400" data-
mol="CC[C@H](C)[C@H]1C(=O)N[C@H]2CSSC[C@@H](C(=O)N[C@@H](CSSC[C@@H](C(
=O)NCC(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)
N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CS
SC[C@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@
H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC2=O)CO)CC(C)C)CC3=CC=C(C=C
3)O)CCC(=O)N)CC(C)C)CCC(=O)O)CC(=O)N)CC4=CC=C(C=C4)O)C(=O)N[C@@H](CC(=
O)N)C(=O)O)C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N
CC(=O)N[C@@H](CC5=CC=CC=C5)C(=O)N[C@@H](CC6=CC=CC=C6)C(=O)N[C@@H](CC7=
CC=C(C=C7)O)C(=O)N[C@@H]([C@@H](C)O)C(=O)N8CCC[C@H]8C(=O)N[C@@H](CCCCN
)C(=O)N[C@@H]([C@@H](C)O)C(=O)O)C(C)C)CC(C)C)CC9=CC=C(C=C9)O)CC(C)C)C)
CCC(=O)O)C(C)C)CC(C)C)CC2=CNC=N2)CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC2
=CNC=N2)NC(=O)[C@H](CCC(=O)N)NC(=O)[C@H](CC(=O)N)NC(=O)[C@H](C(C)C)NC(
=O)[C@H](CC2=CC=CC=C2)N)C(=O)N[C@H](C(=O)N[C@H](C(=O)N1)CO)[C@@H](C)O)
NC(=O)[C@H](CCC(=O)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)[C@H
]([C@@H](C)CC)NC(=O)CN" data-recompute2d data-use-svg></div>
```

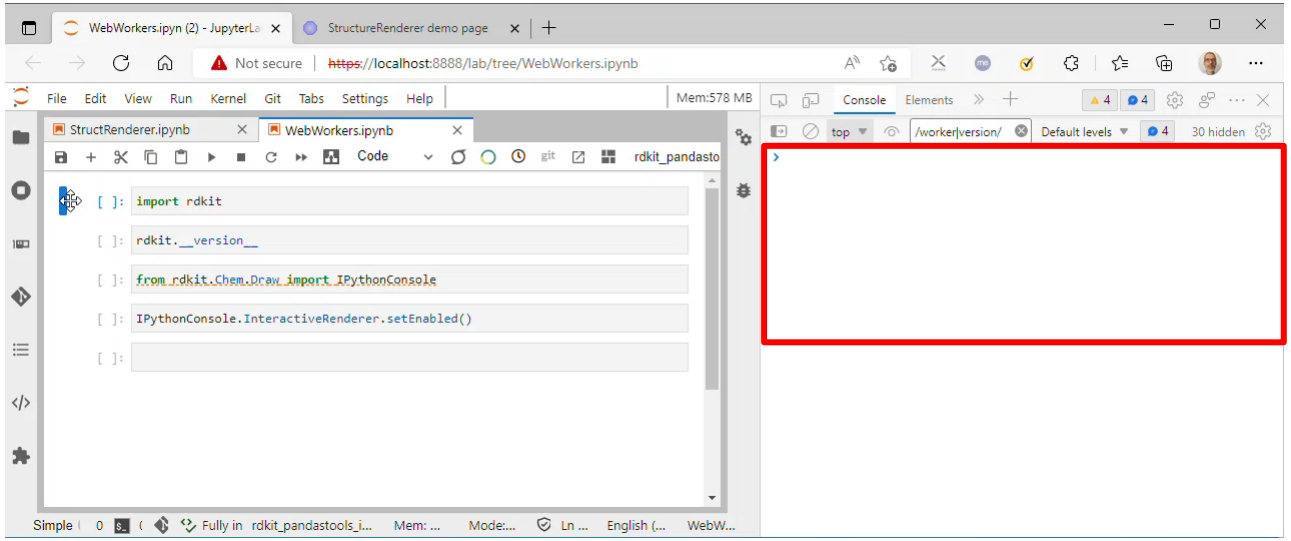


Human insulin

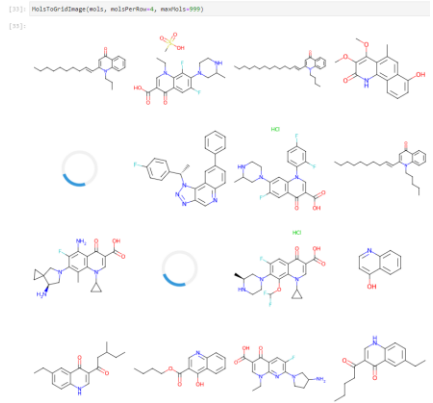
Where we are (2022)

Web Workers!

Web Workers are started as soon as `rdkit-structure-renderer` loads



2D layout jobs are submitted to Web Workers



UI submits 2D layout requests to Scheduler

Scheduler

Scheduler maintains and monitors a queue

Queue

Job11
Job10
Job9
Job8
Job7

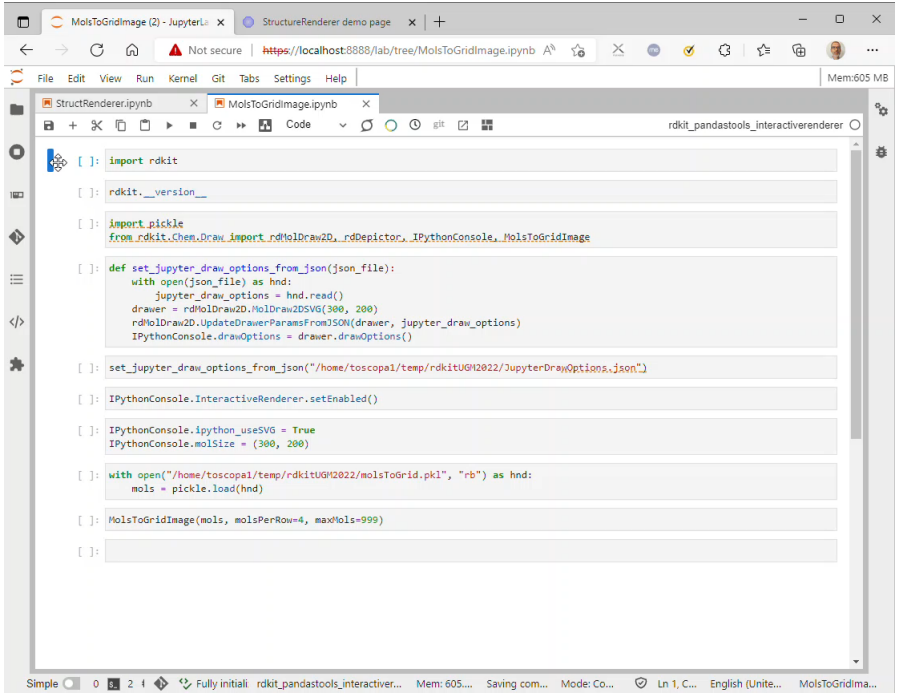
Jobs are scheduled to idle workers

Job3
Job4
Job5
Job6

Worker1 (idle)
Worker2 (Job1)
Worker3 (idle)
Worker4 (Job2)
Worker5 (idle)
Worker6 (idle)

Long running jobs are not an issue

- The UI remains responsive also in the presence of long running jobs as there are multiple Web Workers
- Structures are displayed asynchronously

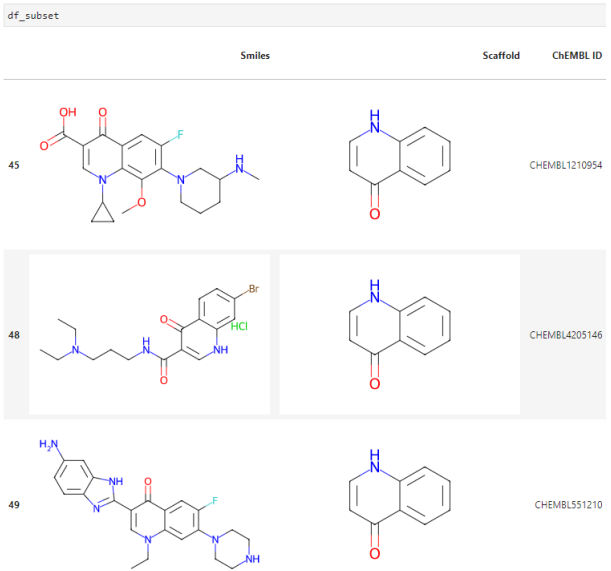


Scaffold alignment (1)

- `mol.generate_aligned_cords(scaffold, opts)`

  This MinimalLib function generates a 2D layout for `mol` by rebuilding side chains *ex novo* around the constrained `scaffold`

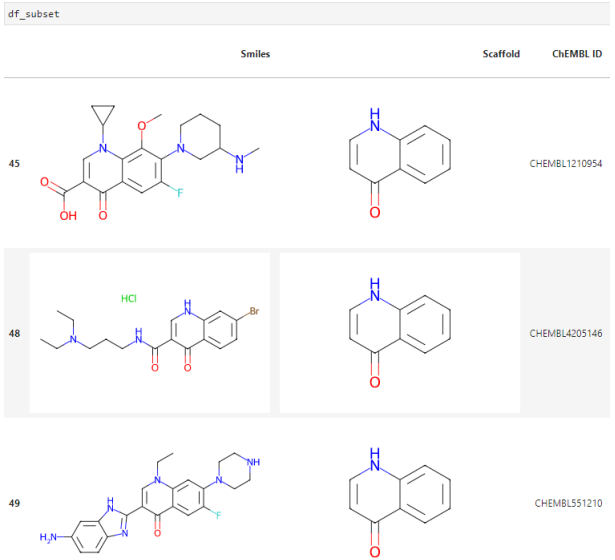
- Based on `RDDepict::generateDepictionMatching2DStructure`



Scaffold alignment (2)

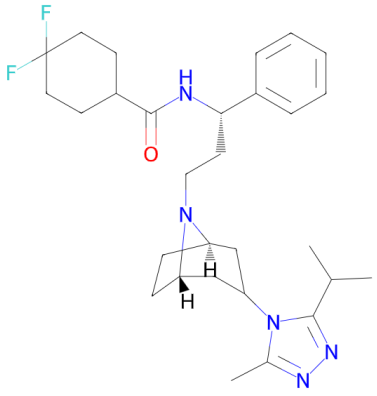
- `JSMol.generate_aligned_cords(scaffold, opts)`

  This MinimalLib function generates a 2D layout for `mol` by rebuilding side chains *ex novo* around the constrained `scaffold`
- Based on `RDDepict::generateDepictionMatching2DStructure`
- Molecules in a chemical series can be oriented consistently

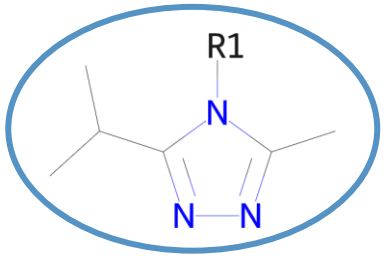


Scaffold alignment (3)

- Rebuilding coordinates *ex novo* may alter the way sidechains were originally drawn

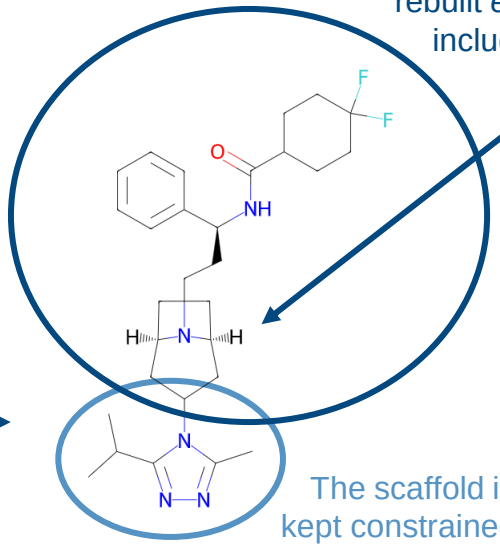


maraviroc



triazole_scaffold

```
maraviroc.generate_aligned_
coords(triazole_scaffold,
'{"useCoordGen":true}')
```

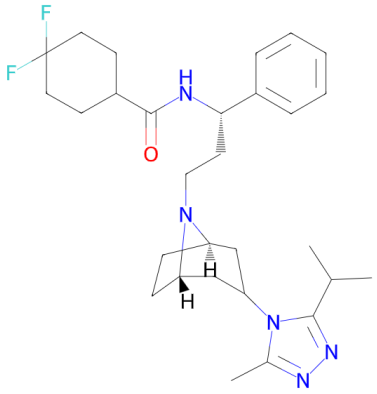


The side chains are rebuilt *ex novo*, including the tropane core

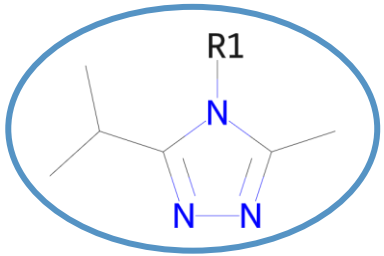
The scaffold is kept constrained

Conservative scaffold alignment (1)

- `JSMol.generate_aligned_cords(scaffold, opts)` now supports an `alignOnly` mode

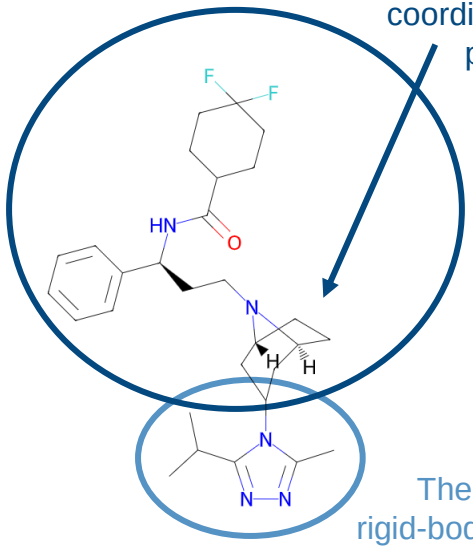


maraviroc



triazole_scaffold

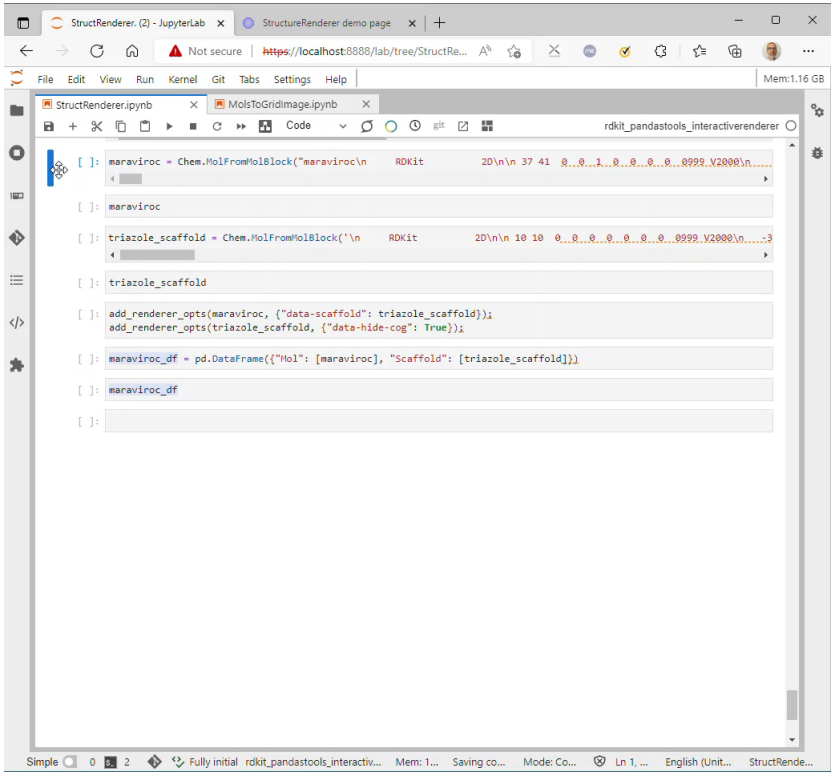
```
maraviroc.generate_aligned_
coords(triazole_scaffold,
'{"useCoordGen":true,
  “alignOnly”:true}}')
```



The original side chain coordinates are preserved

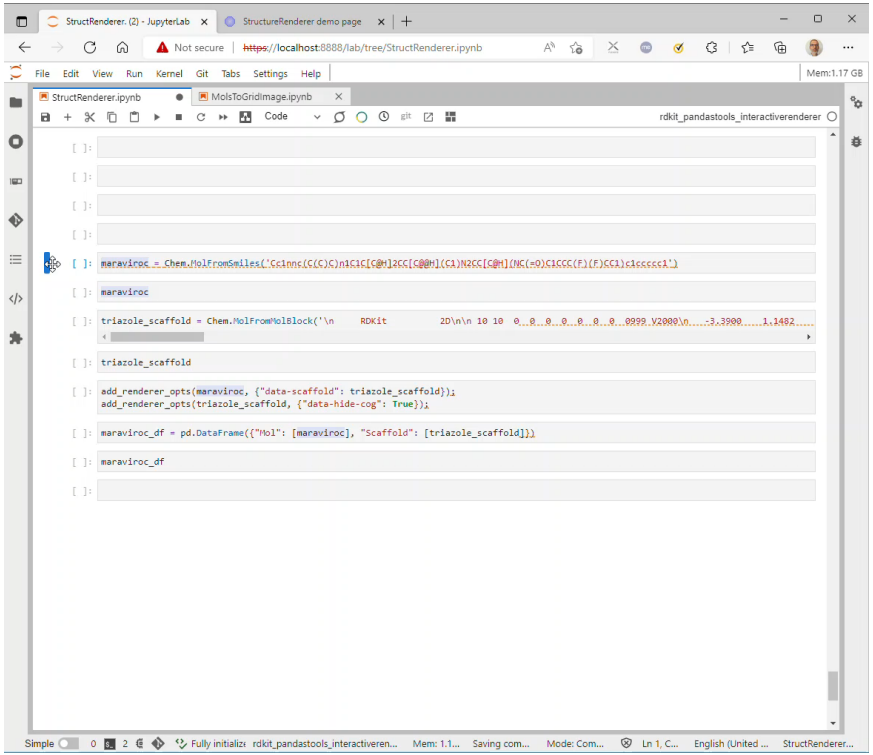
The molecule is rigid-body rotated to best fit the scaffold

Conservative scaffold alignment (2)



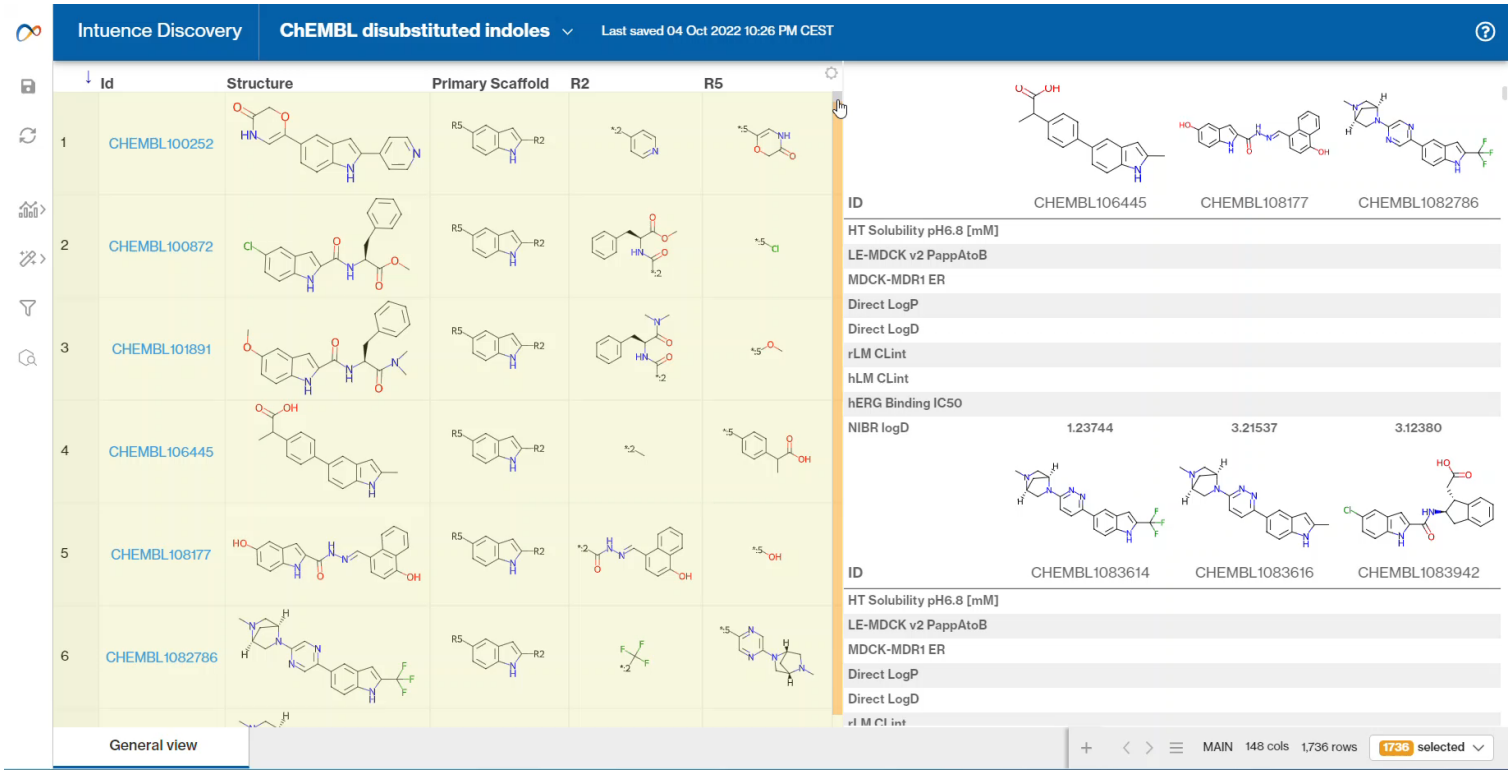
- `rdkit-structure-renderer` will do a conservative scaffold alignment unless explicitly asked to rebuild coordinates *ex novo*

Non-conservative scaffold alignment

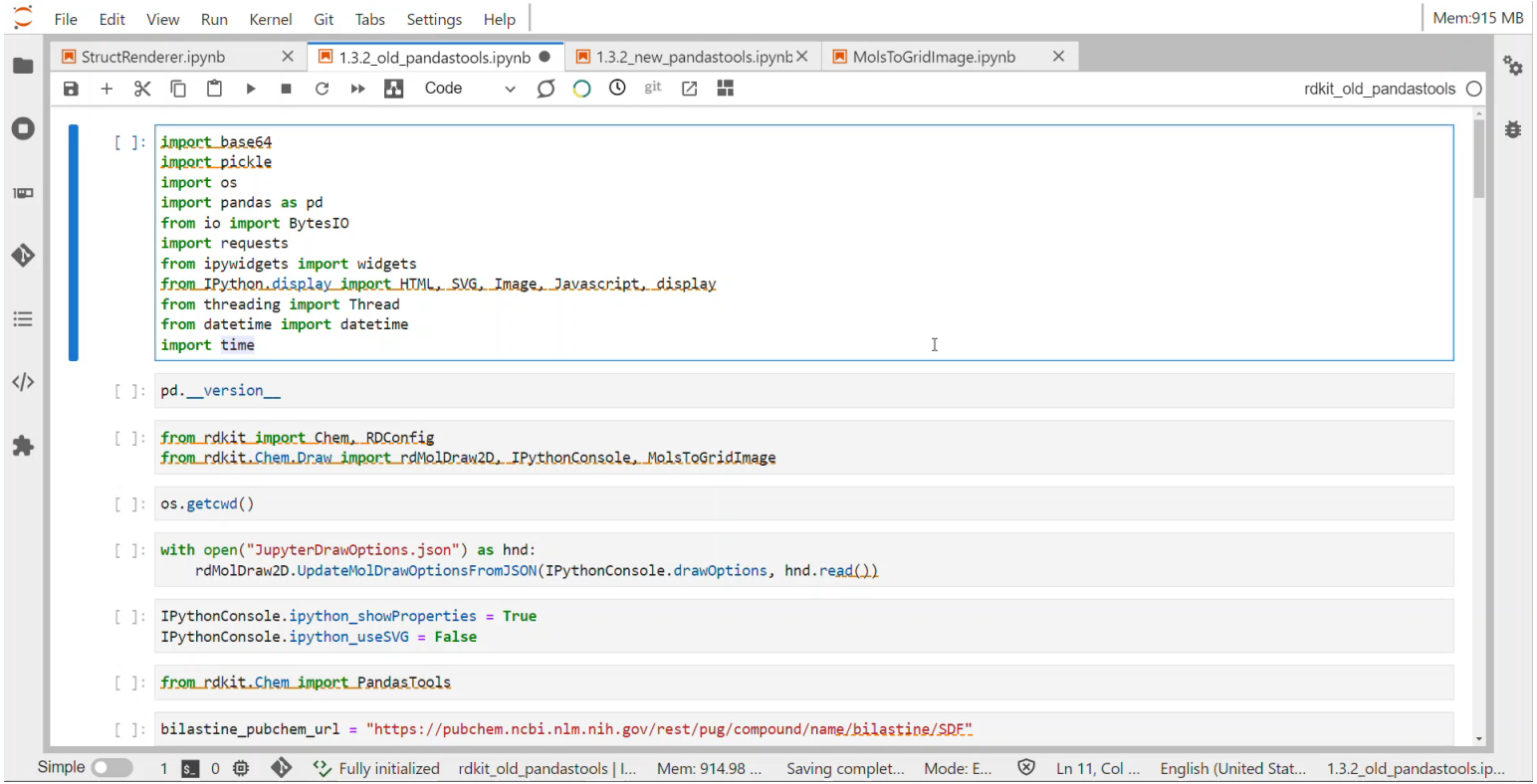


- If the molecule does not have original coordinates, the “Recompute 2D” option is greyed out, since coordinates are anyway rebuilt *ex novo*

NIBR web apps: Intuence Discovery



PandasTools issues (minor)



Jupyter Notebook performance

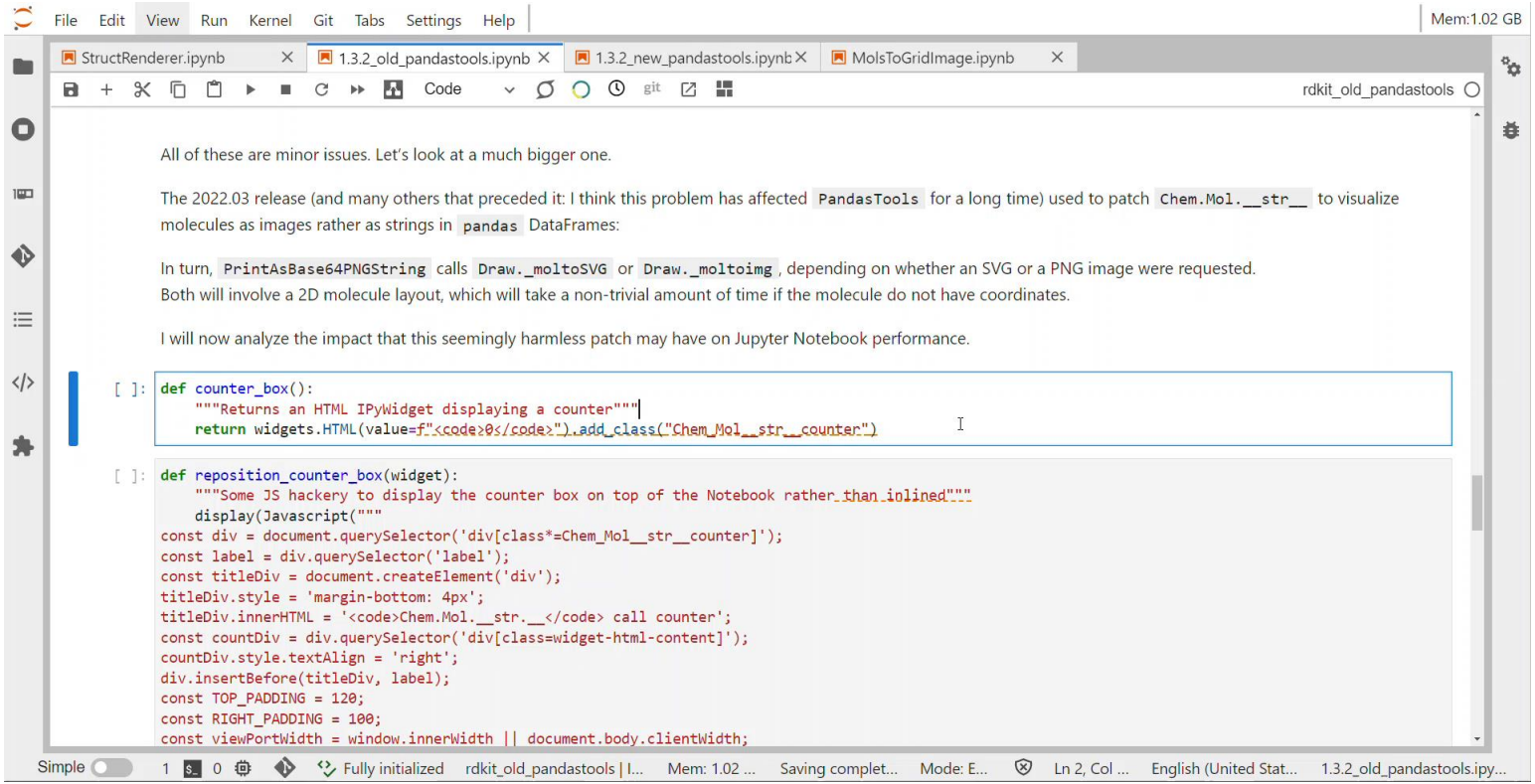
- I have had the feeling for a while that Jupyter Notebooks become sluggish after some uptime
- Google did not provide many relevant hits on that
- I was under the impression that this was happening in particular when working with `pandas` DataFrames
- And my DataFrames always have molecules
- Is `PandasTools` the culprit?

How PandasTools used to work

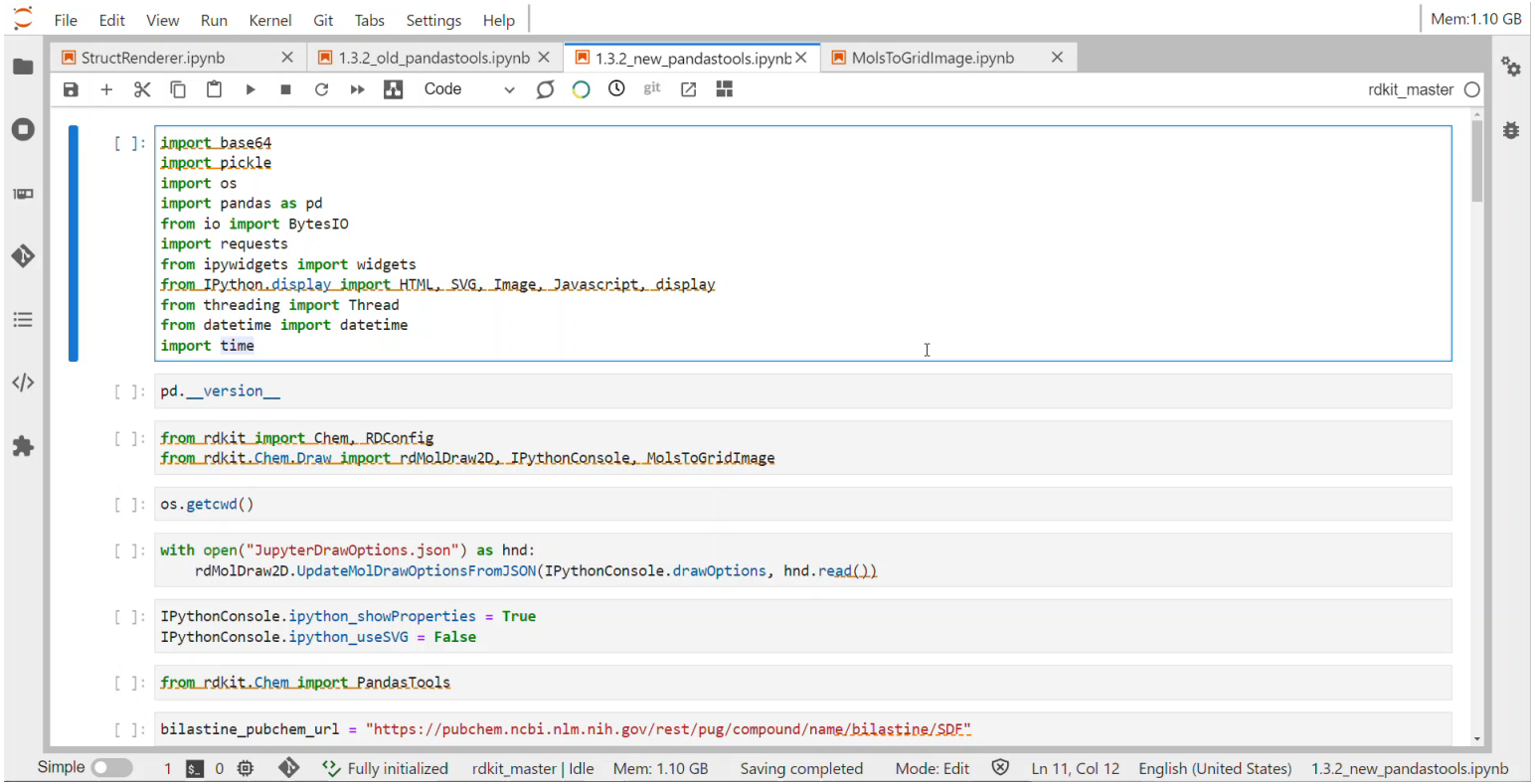
```
469   if renderer == 'String':
470     Chem.Mol.__str__ = PrintDefaultMolRep
471   else:
472     Chem.Mol.__str__ = PrintAsBase64PNGString
473   if frame is not None:
474     frame.to_html = types.MethodType(patchPandasHTMLrepr, frame)
475     if defPandasRepr is not None:
476       frame._repr_html_ = types.MethodType(defPandasRepr, frame)
```

- `Chem.Mol.__str__` is redirected to `PrintAsBase64PNGString` such that molecules are visualized as PNG or SVG images rather than as strings (e.g., <rdkit.Chem.rdchem.Mol object at 0x7fa6bc5441c0>)
- This means that a 2D layout may take place at each `Chem.Mol.__str__` call

PandasTools issues (major)



New PandasTools (using formatters)



What next

Summary

- We have open-sourced on GitHub `rdkit-structure-renderer`, a JS app which enables interactive 2D molecule rendering in the browser
- The only dependency is on RDKit MinimalLib
- Interactive 2D molecule rendering can be enabled in Jupyter Lab with a single line of code and used for visualizing
  - Individual molecules
  - Grids of molecules with `MolsToGridImage`
  - `pandas` DataFrames through `PandasTools`
- We contributed improvements to `PandasTools` and scaffold alignment

Outlook

- Enable built-in magnifying lens functionality
- Enable further interactivity (molecule/atom property visualization, click/hover perception, etc.)
- Extend the interactivity to other environments (KNIME, .NET apps)
- Collect feedback from internal users and external community

Acknowledgments

- NIBR
  - Riccardo Vianello
  - Andreas Liistro
  - Nik Clare
  - Grégori Gerebtzoff
  - Robert Strulak
  - Daniil Tkachev
  - Ekaterina Stepanova
  - Jimmy Kromann
  - [...]
- Datagrok
  - Leonid Stolbov
  - Alex Paramanov
  - Andrew Skalkin
  - [...]
- RDKit
  - Greg Landrum
  - David Cosgrove
  - Brian Kelley
  - [...]
- Schrödinger
  - Nic Zonta
  - Rachel Walker
  - Ricardo Rodríguez
  - [...]

Thank you

NOVARTIS | Reimagining Medicine