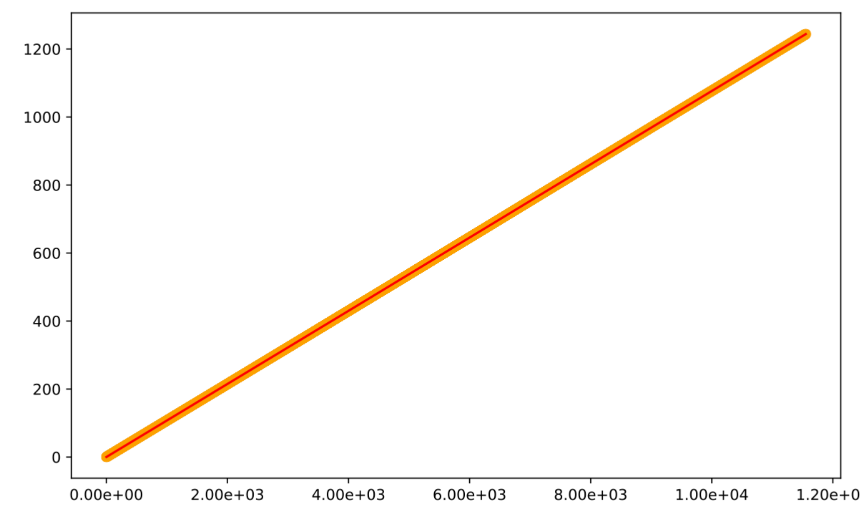
TimsPy: access timsTOF Pro data easily from **Python**

Mateusz K. Łącki[1], Sven Brehmer[2], Ute Distler[1], Stefan Tenzer[1]
[1]University Medical Center of the Johannes Gutenberg University Mainz, [2]Bruker Daltonic GmbH

- timsTOF Pro is a novel instrument by Bruker
- uses trapped ion mobility spectrometry (TIMS) coupled with liquid chromatography (LC) and mass spectrometry (MS)

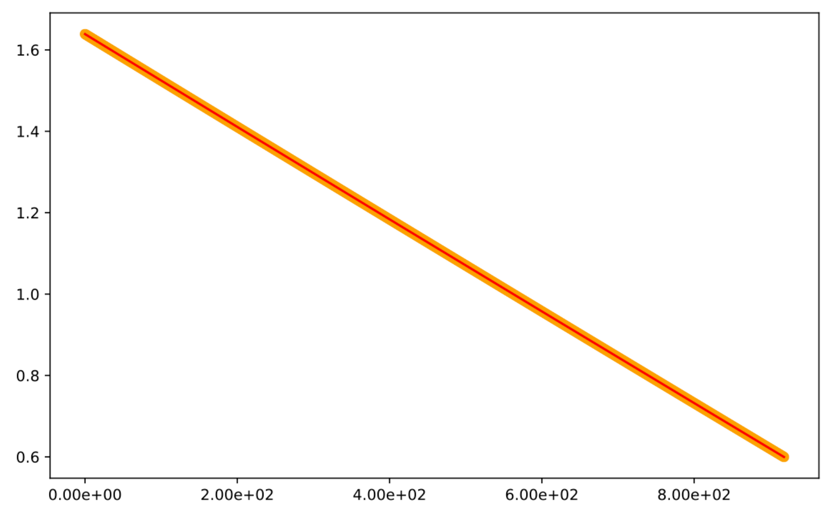
- collected data is stored in vendor's format
  - ★ accessible with freely available software development kit from many programming languages
- data dimensions include:

retention time (RT)



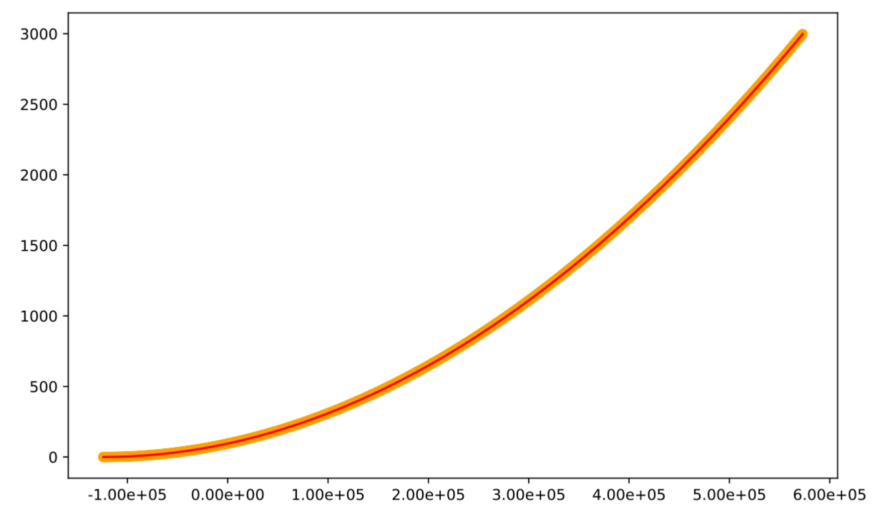
frame number

$1/K_0$ (inverse ion mobility IM)



scan number

mass/charge (MZ)



tof-index

- but are recorded and stored as these numbers.

- **TimsPy** gives you simple access to this data, the way data-scientists love it:

```
from timspy import TimsDIA
D = TimsDIA('path/to/data_folder.d')
D[ frames, scans ]
```

Above, frames select some frames and scans some scans, like on the right:

```
In: D[1:100,0:918]
        frame  scan  tof_idx   i
0           1    33   312260   9
1           1    34   220720   9
2           1    34   261438   9
...       ...   ...      ...  ..
8394       99   914   313598   9
8395       99   917   354548   9
[1626073 rows x 4 columns]
```

Of course, we are talking of gigabytes of data. To minimize RAM usage, **TimsPy** offers iterators: `it = D.iter[ 1:10001, 0:918 ]`

```
In: next(it)
      frame  scan  tof_idx   i
0         1    33   312260   9
1         1    34   220720   9
2         1    34   261438   9
...     ...   ...      ...  ...
1599      1   915   233374   9
1600      1   916   335348  77
[1601 rows x 4 columns]
```

```
In: next(it)
      frame  scan  tof_idx   i
0         2    33    97298   9
1         2    33   310524   9
2         2    34   127985   9
...     ...   ...      ...  ..
6596      2   913    56442   9
6597      2   915   172202   9
[6598 rows x 4 columns]
```

and so on, until 10000th frame is reached

frames and scans can also be very general expressions, covering broad use-cases

```
D[1:5, [33, 50]]
D[(i**2 for i in range(10)), [33, 50]]
D[[1,2,10], 10:100]
D[[1,2,10], [10, 50]]
D['rt > 10 and rt < 50', 1:599]
```

And if you want to directly use physical quantities in the query? We have you covered. Use:

`D.phys[ RT, IM ]` or

`D.physIter[ RT, IM ]`

```
In: D.phys[0.2:10]
            rt        im           mz
0     0.297934  1.601142   302.347671
1     0.297934  1.601142  1165.327281
2     0.297934  1.600000   391.984100
...        ...       ...          ...
8936  9.995394  0.603017   266.728249
8937  9.995394  0.603017  1106.810657
[1554410 rows x 3 columns]
```

We still experiment with usage of VAEX and HDF5 to optimize your daily experience with timsTOF data on your laptop. You don't need a server or a supercomputer to study you data. Follow our github page!

- **Give it a go now! It's as simple as:** `pip install timspy`