dclab Documentation

Release 0.5.0

Paul Müller

May 15, 2018

Contents:

Dclab is a Python library for the post-measurement analysis of real-time deformability cytometry (RT-DC) data sets. This is the documentation of dclab version 0.5.0.

CHAPTER 1

Getting started

1.1 Installing dclab

Dclab depends on several other Python packages:

- fcswrite (.fcs file export),
- h5py (.rtdc file support).
- imageio (.tdms file support, .avi file export),
- nptdms (.tdms file support),
- numpy,
- scipy,
- statsmodels.

In addition, dclab contains code from OpenCV (computation of moments) and scikit-image (computation of contours) to reduce the list of dependencies (these libraries are not required to run dclab).

To install dclab, use one of the following methods (the above package dependencies will be installed automatically):

- **from PyPI:** `pip install dclab`
- **from sources:** `pip install .` or `python setup.py install`

Note that if you are installing from source or if no binary wheel is available for your platform and Python version, Cython will be installed to build the required dclab extensions. If this process fails, please request a binary wheel for your platform (e.g. Windows 64bit) and Python version (e.g. 3.6) by creating a new issue.

1.2 Basic usage

Experimental RT-DC datasets are always loaded with the `new_dataset` method:

```python
import numpy as np
import dclab

# .tdms file format
ds = dclab.new_dataset("/path/to/measurement/Online/M1.tdms")
# .rtdc file format
ds = dclab.new_dataset("/path/to/measurement/M2.rtdc")
```

The object returned by *new_dataset* is always an instance of `RTDCBase`. To show all available features, use:

```python
print(ds.features)
```

This will list all scalar features (e.g. “area_um” and “deform”) and all non-scalar features (e.g. “contour” and “image”). Scalar features can be filtered by editing the configuration of *ds* and calling *ds.apply_filter()*:

```python
# register filtering operations
amin, amax = ds["area_um"].min(), ds["area_um"].max()
ds.config["filtering"]["area_um min"] = (amax + amin) / 2
ds.config["filtering"]["area_um max"] = amax
ds.apply_filter()  # this step is important!
```

This will update the binary array *ds.filter.all* which can be used to extract the filtered data:

```python
area_um_filtered = ds["area_um"][ds.filter.all]
```

It is also possible to create a hierarchy child of this dataset that only contains the filtered data.

```python
ds_child = dclab.new_dataset(ds)
```

The hierarchy child *ds_child* is dynamic, i.e. when the filters in *ds* change, then *ds_child* also changes after calling *ds_child.apply_filter()*.

Non-scalar features do not support fancy indexing (i.e. *ds[“image”][ds.filter.all]* will not work. Use a for-loop to extract them.

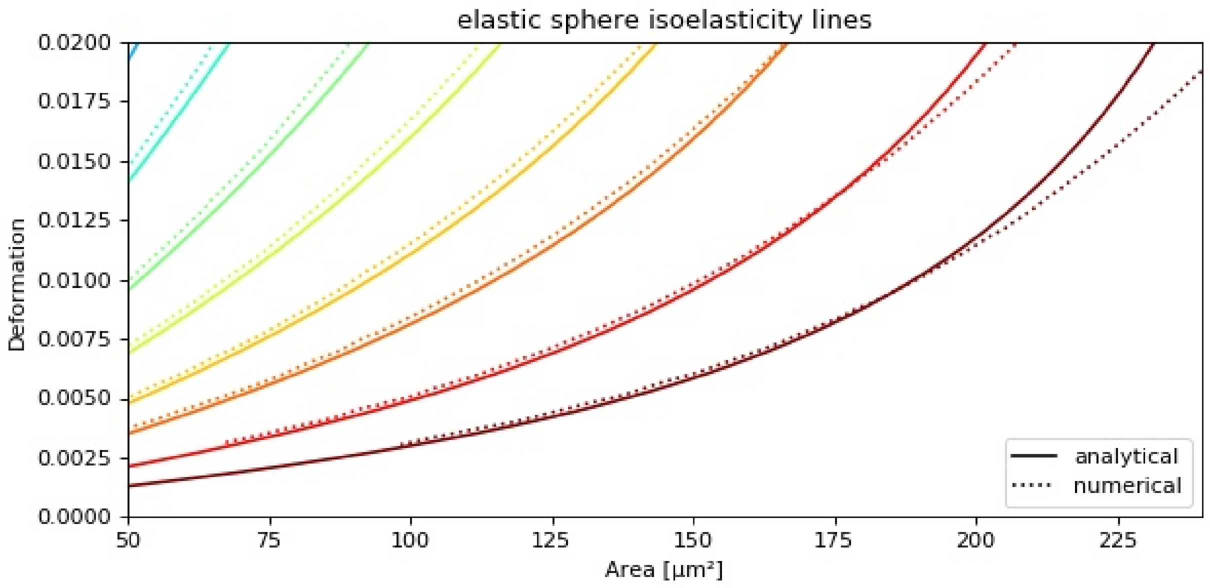
```python
for ii in range(len(ds)):
    image = ds["image"][ii]
    mask = ds["mask"][ii]
    # this is equivalent to ds["bright_avg"][ii]
    bright_avg = np.mean(image[mask])
    print("average brightness of event {}: {:.1f}".format(ii, bright_avg))
```

CHAPTER 2

Examples

2.1 Plotting isoelastics

This example illustrates how to plot dclab isoelastics by reproducing figure 3 (lower left) of [MMM+17].



isoelastics.py

```
import matplotlib.pylab as plt
import matplotlib.lines as mlines
from matplotlib import cm
import numpy as np

```

(continues on next page)

(continued from previous page)

```python
import dclab

# parameters for isoelastics
kwargs = {"col1": "area_um",  # x-axis
          "col2": "deform",  # y-axis
          "channel_width": 20,  # [um]
          "flow_rate": 0.04,  # [ul/s]
          "viscosity": 15,  # [Pa s]
          "add_px_err": False  # no pixelation error
          }

isos = dclab.isoelastics.get_default()
analy = isos.get(method="analytical", **kwargs)
numer = isos.get(method="numerical", **kwargs)

plt.figure(figsize=(8, 4))
ax = plt.subplot(111, title="elastic sphere isoelasticity lines")
colors = [cm.get_cmap("jet")(x) for x in np.linspace(0, 1, len(analy))]
for aa, nn, cc in zip(analy, numer, colors):
    ax.plot(aa[:, 0], aa[:, 1], color=cc)
    ax.plot(nn[:, 0], nn[:, 1], color=cc, ls=":")

line = mlines.Line2D([], [], color='k', label='analytical')
dotted = mlines.Line2D([], [], color='k', ls=":", label='numerical')
ax.legend(handles=[line, dotted])

ax.set_xlim(50, 240)
ax.set_ylim(0, 0.02)
ax.set_xlabel(dclab.dfn.feature_name2label["area_um"])
ax.set_ylabel(dclab.dfn.feature_name2label["deform"])

plt.tight_layout()
plt.show()
```

CHAPTER 3

Command line interface

3.1 tdms2rtdc

Convert RT-DC .tdms files to the hdf5-based .rtdc file format. Note: Do not delete original .tdms files after conversion. The conversion might be incomplete.

```
usage: dclab-tdms2rtdc [-h] [--compute-ancillary-features] tdms-path rtdc-path
```

3.1.1 Positional Arguments

tdms-path	Input path (tdms file or folder containing tdms files)
rtdc-path	Output path (file or folder), existing data will be overridden

3.1.2 Named Arguments

--compute-ancillary-features Compute features, such as volume or emodulus, that are otherwise computed on-the-fly. Use this if you want to minimize analysis time in e.g. ShapeOut. CAUTION: ancillary feature recipes might be subject to change (e.g. if an error is found in the recipe). Disabling this option maximizes compatibility with future versions and allows to isolate the original data.

Default: False

3.2 verify-dataset

Check experimental data sets for completeness. Note that old measurements will most likely fail this verification step. This program is used to enforce data integrity with future implementations of RT-DC recording software (e.g. ShapeIn).

```
usage: dclab-verify-dataset [-h] path
```

3.2.1 Positional Arguments

path	Path to experimental dataset

CHAPTER 4

Code reference

4.1 definitions

Naming conventions

4.2 downsampling

Content-based downsampling of ndarrays

`dclab.downsampling.`**`downsample_rand`**`(`*a*, *samples*, *remove_invalid=True*, *retidx=False*`)`

Downsampling by randomly removing points

Parameters

- **`a`** (*`1d ndarray`*) – The input array to downsample
- **`samples`** (*`int`*) – The desired number of samples
- **`remove_invalid`** (*`bool`*) – Remove nan and inf values before downsampling
- **`retidx`** (*`bool`*) – Also return a boolean array that corresponds to the downsampled indices in *a*.

Returns

- **dsa, dsb** (*1d ndarrays of shape (samples,)*) – The pseudo-randomly downsampled arrays *a* and *b*
- **[idx]** (1d boolean array with same shape as *a*) – A boolean array such that *a[idx] == dsa* is all true

4.3 features

Basic methods for event feature computation

4.4 isoelastics

Isoelastics management

class dclab.isoelastics.**Isoelastics**(*paths=[]*)

add(*isoel*, *col1*, *col2*, *channel_width*, *flow_rate*, *viscosity*, *method*)
Add isoelastics

Parameters

- **isoel** (*list of ndarrays*) – Each list item resembles one isoelastic line stored as an array of shape (N,3). The last column contains the emodulus data.
- **col1** (*str*) – Name of the first feature of all isoelastics (e.g. isoel[0][:,0])
- **col2** (*str*) – Name of the second feature of all isoelastics (e.g. isoel[0][:,1])
- **channel_width** (*float*) – Channel width in µm
- **flow_rate** (*float*) – Flow rate through the channel in µl/s
- **viscosity** (*float*) – Viscosity of the medium in mPa*s
- **method** (*str*) – The method used to compute the isoelastics (must be one of *VALID_METHODS*).

Notes

The following isoelastics are automatically added for user convenience: - isoelastics with *col1* and *col2* interchanged - isoelastics for circularity if deformation was given

static add_px_err(*isoel*, *col1*, *col2*, *px_um*, *inplace=False*)
Undo pixelation correction

Isoelasticity lines are already corrected for pixelation effects as described in

Mapping of Deformation to Apparent Young's Modulus in Real-Time Deformability Cytometry Christoph Herold, arXiv:1704.00572 [cond-mat.soft] (2017) https://arxiv.org/abs/1704.00572.

If the isoealsticity lines are displayed with deformation data that are not corrected, then the lines must be "un"-corrected, i.e. the pixelation error must be added to the lines to match the experimental data.

Parameters

- **isoel** (*list of 2d ndarrays of shape (N, 3)*) – Each item in the list corresponds to one isoelasticity line. The first column is defined by *col1*, the second by *col2*, and the third column is the emodulus.
- **col2** (*col1,*) – Define the fist to columns of each isoelasticity line. One of ["area_um", "circ", "deform"]
- **px_um** (*float*) – Pixel size [µm]

static check_col12(*col1*, *col2*)

static convert(*isoel*, *col1*, *col2*, *channel_width_in*, *channel_width_out*, *flow_rate_in*, *flow_rate_out*, *viscosity_in*, *viscosity_out*, *inplace=False*)
Convert isoelastics in area_um-deform space

Parameters

- **isoel** (*list of 2d ndarrays of shape (N, 3)*) – Each item in the list corresponds to one isoelasticity line. The first column is defined by *col1*, the second by *col2*, and the third column is the emodulus.
- **col2** (*col1,*) – Define the fist to columns of each isoelasticity line. One of ["area_um", "circ", "deform"]
- **channel_width_in** (*float*) – Original channel width [µm]
- **channel_width_out** (*float*) – Target channel width [µm]
- **flow_rate_in** (*float*) – Original flow rate [µl/s]
- **flow_rate_in** – Target flow rate [µl/s]
- **viscosity_in** (*float*) – Original viscosity [mPa*s]
- **viscosity_out** (*float*) – Target viscosity [mPa*s]

Notes

If only the positions of the isoelastics are of interest and not the value of the elastic modulus, then it is sufficient to supply values for the channel width and set the values for flow rate and viscosity to a constant (e.g. 1).

See also:

`dclab.features.emodulus.convert()` conversion method used

get(*col1*, *col2*, *method*, *channel_width*, *flow_rate=None*, *viscosity=None*, *add_px_err=False*, *px_um=None*)
Get isoelastics

Parameters

- **col1** (*str*) – Name of the first feature of all isoelastics (e.g. isoel[0][:,0])
- **col2** (*str*) – Name of the second feature of all isoelastics (e.g. isoel[0][:,1])
- **method** (*str*) – The method used to compute the isoelastics (must be one of *VALID_METHODS*).
- **channel_width** (*float*) – Channel width in µm
- **flow_rate** (float or *None*) – Flow rate through the channel in µl/s. If set to *None*, the flow rate of the imported data will be used (only do this if you do not need the correct values for elastic moduli).
- **viscosity** (float or *None*) – Viscosity of the medium in mPa*s. If set to *None*, the flow rate of the imported data will be used (only do this if you do not need the correct values for elastic moduli).
- **add_px_err** (*bool*) – If True, add pixelation errors according to C. Herold (2017), https://arxiv.org/abs/1704.00572
- **px_um** (*float*) – Pixel size [µm], used for pixelation error computation

See also:

`dclab.features.emodulus.convert()` conversion in-between channel sizes and viscosities

`dclab.features.emodulus.corrpix_deform_delta()` pixelation error that is applied to the deformation data

`load_data`(*path*)
Load isoelastics from a text file

The text file is loaded with *numpy.loadtxt* and must have three columns, representing the two data columns and the elastic modulus with units defined in *definitions.py*. The file header must have a section defining meta data of the content like so:

[. . .] # # - column 1: area_um # - column 2: deform # - column 3: emodulus # - channel width [um]: 20 # - flow rate [ul/s]: 0.04 # - viscosity [mPa*s]: 15 # - method: analytical # # [. . .]

Parameters **`path`** (*`str`*) – Path to a isoelastics text file

class `dclab.isoelastics.`**`IsoelasticsDict`**

`dclab.isoelastics.`**`get_default`**()
Return default isoelasticity lines

4.5 kde_methods

Kernel Density Estimation methods

`dclab.kde_methods.`**`get_bad_vals`**(*x*, *y*)

`dclab.kde_methods.`**`ignore_nan_inf`**(*kde_method*)
Ignores nans and infs from the input data

Invalid positions in the resulting density are set to nan.

`dclab.kde_methods.`**`kde_gauss`**(*events_x*, *events_y*, *xout=None*, *yout=None*, **args*, ***kwargs*)
Gaussian Kernel Density Estimation

Parameters

- **`events_y`** (*`events_x,`*) – The input points for kernel density estimation. Input is flattened automatically.
- **`yout`** (*`xout,`*) – The coordinates at which the KDE should be computed. If set to none, input coordinates are used.

Returns **density** – The KDE for the points in (xout, yout)

Return type ndarray, same shape as *xout*

See also:

scipy.stats.gaussian_kde

Notes

This is a wrapped version that ignores nan and inf values.

`dclab.kde_methods.`**`kde_histogram`**`(`*events_x*, *events_y*, *xout=None*, *yout=None*, **args*, ***kwargs*`)`

Histogram-based Kernel Density Estimation

Parameters

- **`events_y`** (*`events_x,`*) – The input points for kernel density estimation. Input is flattened automatically.
- **`yout`** (*`xout,`*) – The coordinates at which the KDE should be computed. If set to none, input coordinates are used.
- **`bins`** (*`tuple (binsx, binsy)`*) – The number of bins to use for the histogram.

Returns **density** – The KDE for the points in (xout, yout)

Return type ndarray, same shape as *xout*

See also:

numpy.histogram2d scipy.interpolate.RectBivariateSpline

Notes

This is a wrapped version that ignores nan and inf values.

`dclab.kde_methods.`**`kde_none`**`(`*events_x*, *events_y*, *xout=None*, *yout=None*`)`

No Kernel Density Estimation

Parameters

- **`events_y`** (*`events_x,`*) – The input points for kernel density estimation. Input is flattened automatically.
- **`yout`** (*`xout,`*) – The coordinates at which the KDE should be computed. If set to none, input coordinates are used.

Returns **density** – The KDE for the points in (xout, yout)

Return type ndarray, same shape as *xout*

Notes

This method is a convenience method that always returns ones in the shape that the other methods in this module produce.

`dclab.kde_methods.`**`kde_multivariate`**`(`*events_x*, *events_y*, *xout=None*, *yout=None*, **args*, ***kwargs*`)`

Multivariate Kernel Density Estimation

Parameters

- **`events_y`** (*`events_x,`*) – The input points for kernel density estimation. Input is flattened automatically.
- **`bw`** (*`tuple (bwx, bwy) or None`*) – The bandwith for kernel density estimation.
- **`yout`** (*`xout,`*) – The coordinates at which the KDE should be computed. If set to none, input coordinates are used.

Returns **density** – The KDE for the points in (xout, yout)

Return type ndarray, same shape as *xout*

See also:

statsmodels.nonparametric.kernel_density.KDEMultivariate

Notes

This is a wrapped version that ignores nan and inf values.

4.6 polygon_filter

PolygonFilter classes and methods

class `dclab.polygon_filter.`**`PolygonFilter`**(*axes=None*, *points=None*, *inverted=False*, *name=None*, *filename=None*, *fileid=0*, *unique_id=None*)

An object for filtering RTDC data based on a polygonial area

`instances = []`

`static clear_all_filters`()

Remove all filters and reset instance counter

`copy`(*invert=False*)

Return a copy of the current instance

Parameters **`invert`** (`bool`) – The copy will be inverted w.r.t. the original

`filter`(*datax*, *datay*)

Filter a set of datax and datay according to *self.points*

`static get_instance_from_id`(*unique_id*)

Get an instance of the *PolygonFilter* using a unique id

`static import_all`(*path*)

Import all polygons from a .poly file.

Returns a list of the imported polygon filters

`static instace_exists`(*unique_id*)

Determine whether an instance with this unique id exists

`static point_in_poly`(*x*, *y*, *poly*)

Determine whether a point is within a polygon area

Parameters

- **`y`** (`x,`) – The coordinates of the point
- **`poly`** (`list-like`) – The polygon (*PolygonFilter.points*)

Returns **inside** – *True*, if point is inside.

Return type bool

`static remove`(*unique_id*)

Remove a polygon filter from *PolygonFilter.instances*

`save`(*polyfile*, *ret_fobj=False*)

Save all data to a text file (appends data if file exists).

Polyfile can be either a path to a file or a file object that was opened with the write "w" parameter. By using the file object, multiple instances of this class can write their data.

If *ret_fobj* is *True*, then the file object will not be closed and returned.

static save_all(*polyfile*)
Save all polygon filters

exception `dclab.polygon_filter.`**PolygonFilterError**

`dclab.polygon_filter.`**get_polygon_filter_names**()
Get the names of all polygon filters in the order of creation

4.7 rtdc_dataset

4.8 statistics

Statistics computation for RT-DC dataset instances

class `dclab.statistics.`**Statistics**(*name*, *method*, *req_feature=False*)

available_methods = {'%-gated': <dclab.statistics.Statistics object at 0x7f592cf09eb8

get_feature(*rtdc_ds*, *axis*)

get_data(*kwargs*)

`dclab.statistics.`**flow_rate**(*mm*)

`dclab.statistics.`**get_statistics**(*rtdc_ds*, *methods=None*, *features=None*)

Parameters

- **rtdc_ds** (instance of *dclab.rtdc_dataset.RTDCBase*.) – The data set for which to compute the statistics.
- **methods** (`list of str or None`) – The methods wih which to compute the statistics. The list of available methods is given with *dclab.statistics.Statistics.available_methods.keys()* If set to *None*, statistics for all methods are computed.
- **features** (`list of str`) – Feature name identifiers are defined in *dclab.definitions.scalar_feature_names*. If set to *None*, statistics for all axes are computed.

Returns

- **header** (*list of str*) – The header (feature + method names) of the computed statistics.
- **values** (*list of float*) – The computed statistics.

`dclab.statistics.`**mode**(*data*)
Compute an intelligent value for the mode

The most common value in experimental is not very useful if there are a lot of digits after the comma. This method approaches this issue by rounding to bin size that is determined by the Freedman–Diaconis rule.

Parameters **data** (`1d ndarray`) – The data for which the mode should be computed.

Returns **mode** – The mode computed with the Freedman-Diaconis rule.

Return type float

CHAPTER 5

Bilbliography

CHAPTER 6

Imprint/Impressum

6.1 Imprint and disclaimer

For more information, please refer to the imprint and disclaimer (Impressum und Haftungsausschluss) at https://www.zellmechanik.com/Imprint.html.

6.2 Privacy policy

This documentation is hosted on https://readthedocs.org/ whose privacy policy applies (see issue #2602).

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Bibliography

[MMM+17] M. Mokbel, D. Mokbel, A. Mietke, N. Träber, S. Girardo, O. Otto, J. Guck, and S. Aland. Numerical simulation of real-time deformability cytometry to extract cell mechanical properties. *ACS Biomaterials Science & Engineering*, 3(11):2962–2973, jan 2017. doi:10.1021/acsbiomaterials.6b00558.

Python Module Index

Index