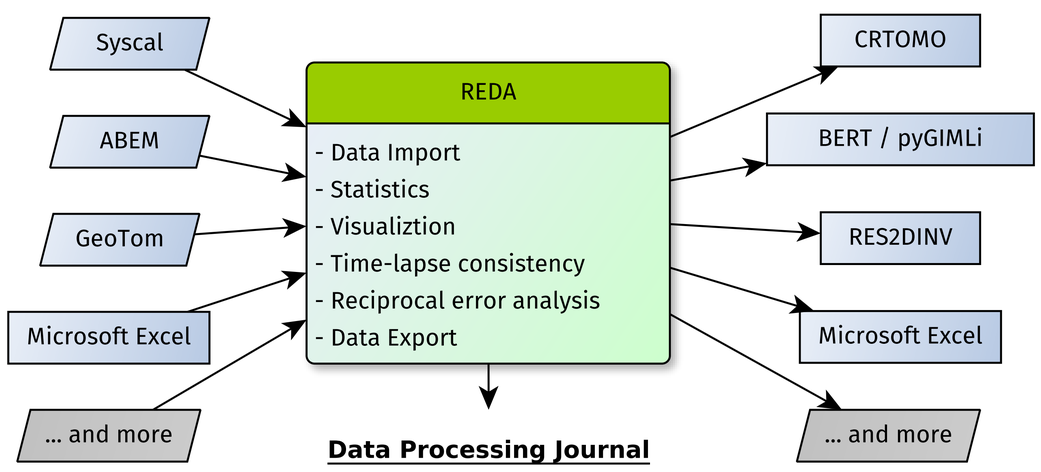
Towards unified and reproducible processing of geoelectrical data

Maximilian Weigand & Florian M. Wagner

University of Bonn, Steinmann Institute, Department of Geophysics

REDA - A Python package for reproducible electrical data analysis

REDA is a scientific Python library for reproducible geoelectrical data analysis. It aims to provide a unified interface for common and advanced data processing steps while bridging the gap between a multitude of geoelectric measurement devices and inversion codes used across the geophysical community. It offers functionality to import, analyze, process, visualize, and export geoelectrical data with particular emphasis on time-lapse functionality and reproducibility. REDA is open-source under the permissive MIT license. We look forward to your contribution!



Built on the powerful *pandas.DataFrame*

REDA is built upon the flexible pandas.DataFrame container, which allows straightforward manipulation, analysis, and interpretation of categorical multi-dimensional data. While convenience functions are provided, access to the DataFrame is easily available.

Time-lapse analysis

DataFrames allow us to apply processing steps just to certain time steps. Error models and normal-reciprocal analysis are easily extended to the time axis.

Filtering never was easier!

Using simplified programming statements, apply complex filters to the data:

```
data.filter('K  > 1000')
```

Apply filters to subsets of the data:

```
data.sub_filter(subset='timestep in [10, 80]',
                filter='A > 10 and > A < 20 and K > 1000')
```

The Data Journal

Use the Data Journal to keep track of all actions applied to the data in human-readable form. The effect of each filter is recorded in the journal, helping to establish reproducibility for your analysis.

Conclusions

- REDA can simplify your data analysis and make it reproducible for others independent of the device and inversion code used.
- REDA harnesses the power of Python, Jupyter, and pandas.
- REDA is free, open-source, platform compatible, tested and welcomes users and contributors.

Try it on the web today! (Or install on your computer + +)

Scan the QR code to open the full example as a Jupyter Notebook in your browser or enter the url:

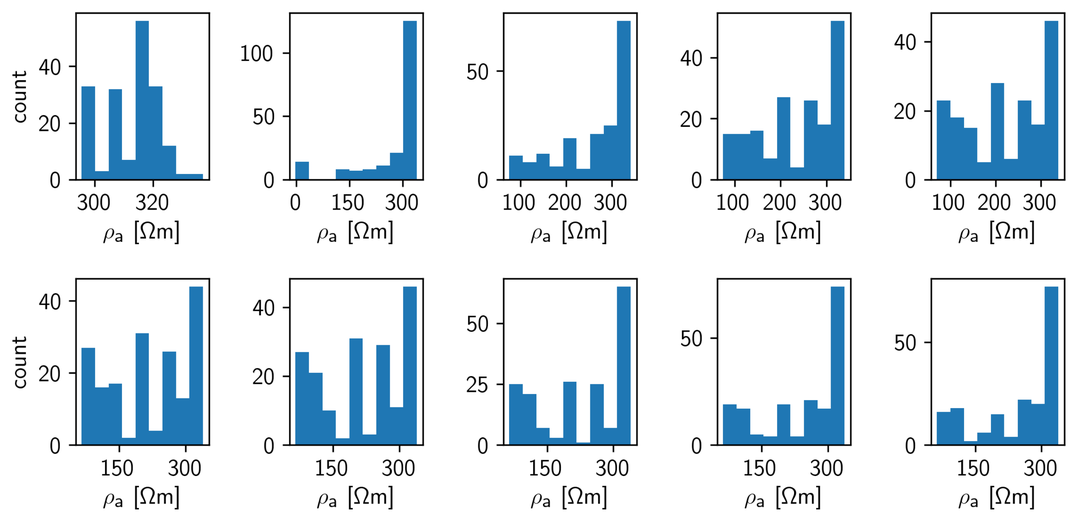
http://tinyurl.com/yauesj5t

</> Workflow demonstration

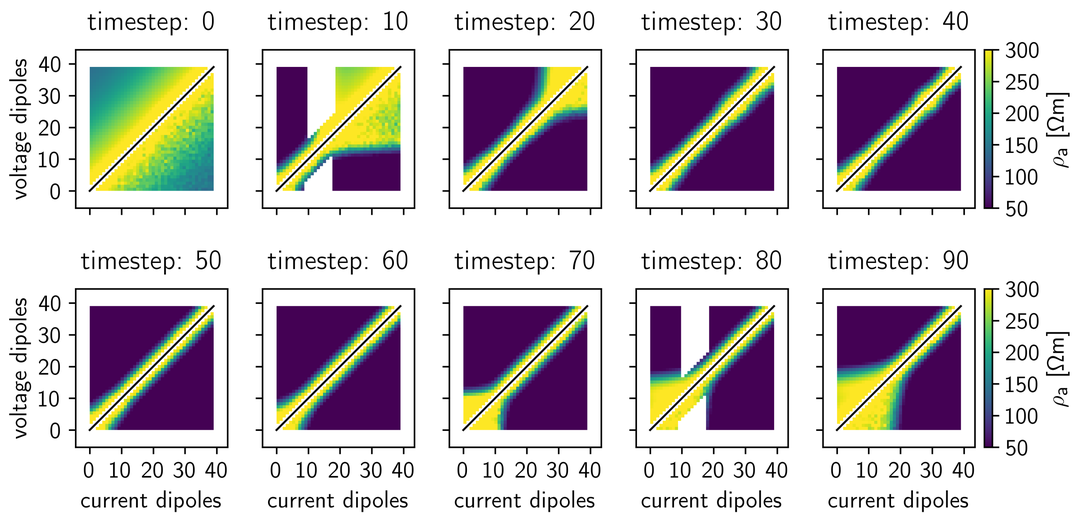
```
# import time steps
from reda import ERT
data = ERT()
for nr, filename in enumerate(filenames):
    data.import_bert(filename, timestep=nr)
ERT.assign_norrec()
```

timestep	A	B	M	N	norrec	id	K [m]	R [Ω]	ρ_a [Ωm]
1	1	2	4	3	nor	111	18.84	16.69	314.68
1	4	3	1	2	rec	111	18.84	16.69	314.68
1	1	2	5	4	nor	147	75.39	4.23	319.55
1	5	4	1	2	rec	147	75.39	4.23	319.55
...	...	...	...	..	...	...	...	...	...

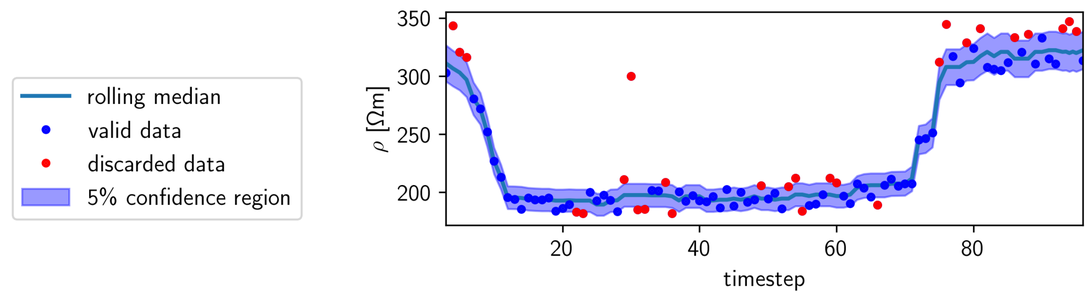
```
# plot histograms, use filter queries to select timesteps
import reda.plotters.histograms as RH
results = RH.plot_histograms_extra_dims(
data, keys = ['rho_a'], extra_dims=['timestep'], Nx=5,
    subquery='timestep in [0, 10, 20, 30, 40, 50, 60, 70, 80, 90]')
```




```
# plot pseudosections:
import reda.plotters.pseudoplots as PS
fig = PS.plot_ps_extra(data, 'rho_a',
    subquery='timestep in [0, 10, 20, 30, 40, 50, 60, 70, 80, 90]'
)
```




```
# use a rolling median filter to detect outliers:
import reda.plotters.time_series as TS
fig, ax = TS.plot_quadpole_evolution(data, quadpole=[10, 11, 15, 14],
    columns=['rho_a', ], threshold=0.05, rolling=True, )
```




```
# print the journal related to all data activities
data.print_data_journal()
--- Data Journal Start ---
2017-11-15 12:26:57.659537
Data was imported from file pygimli_paper_data000.ohm (741 data points)
...
Data was imported from file pygimli_paper_data099.ohm (741 data points)
A filter was applied with query "K > 1000". In total 55934 records were
removed.
--- Data Journal End ---
```

```
ert.export_bert(directory='output_bert')
```

mweigand@geo.uni-bonn.de & wagner@geo.uni-bonn.de

github.com/geophysics-ubonn/reda