

Standardization of data ontologies and research data management

Michael Götte
Research Data Steward
Helmholtz Zentrum Berlin

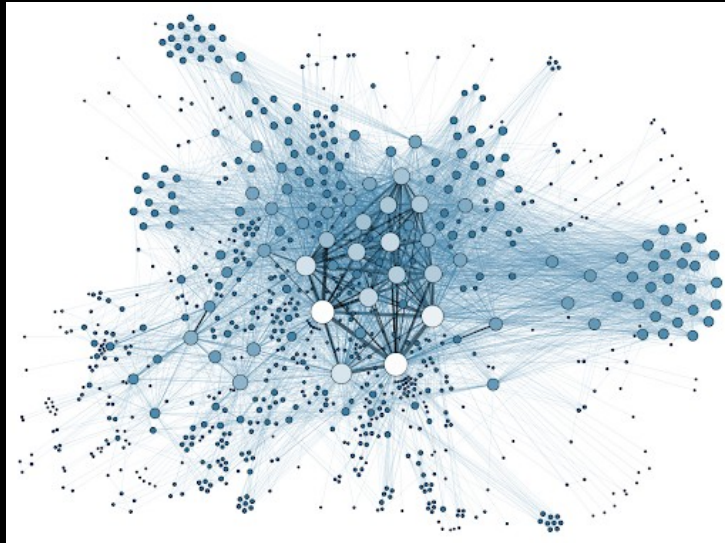
The principles of Linked Data

(slide from <https://jakub.klimek.com/nswi144>)

- 1) Use URIs as names for things.
- 2) Use HTTP URIs (URLs) so that people can look up those names.
- 3) When someone looks up a URI/URL, provide useful information.
- 4) Include links to other URIs/URLs so that they can discover more things.

Linked Open Data (LOD)

- No standard file formats
- Data can be stored in files, databases, web pages, needs to be „linkable“
- Context is provided through links to other entities (e.g. Ontology class definitions)



See: <https://jakub.klimek.com/nswi144> for some lecture notes

Exploring semantics

Published (alpha) thin film solar cell ontology:

<https://matportal.org/ontologies/TFSCO/?p=summary>

We use this ontology to connect Nomad entities with the semantic:

```
class SpinCoating(WetChemicalDeposition):  
    '''Base class for spin coating of a sample'''  
    m_def = Section(  
        #Link to ontology class 'spin coating'  
        links = ['http://purl.obolibrary.org/obo/CHMO\_0001472']  
    )
```

http://purl.obolibrary.org/obo/CHMO_0001472

https://purl.archive.org/tfSCO/TFSCO_00005034

```
slot_die_head_distance_to_thinfil = Quantity(  
    # Link to ontology class 'slot die head distance to thinfilm' and 'slot die head distane to thinfilm setting datum'  
    links=['http://www.semanticweb.org/ot2661/ontologies/2022/8/TFSCO#TFSCO\_00005034',  
          'http://www.semanticweb.org/ot2661/ontologies/2022/8/TFSCO#TFSCO\_00005044'],  
    type=np.dtype(  
        np.float64),  
    unit=('mm'),  
    a_eIn=dict(  
        component='NumberEditQuantity',  
        defaultDisplayUnit='mm',  
        props=dict(  
            minValue=0)))
```

Showcase in Nomad on Data annotations

https://nomad-hzb-se.de/nomad-oasis/gui/user/uploads/upload/id/JQr0kd9SQmevZ_BF_JGlcg/entry/id/XLMy_MNle_hJrfOZ_zWbOyB_6cvt/data/results/properties/optoelectronic/solar_cell/efficiency

Go to Nomad (<https://nomad-hzb-se.de/nomad-oasis/gui/about/information>)

Showcase in Nomad on Data annotations (screen shots)

The screenshot displays the Nomad web interface for data annotations. The top navigation bar includes 'PUBLISH', 'EXPLORE', 'ANALYZE', and 'ABOUT'. The user is logged in as 'Michael Götte'. The main content area is divided into four panels:

- Overview:** Shows a search bar and a plot of Current Density (A/m²) vs Voltage (V). The plot displays several curves for forward and reverse current densities under different light conditions.
- Files:** Shows a tree view of data files, with 'HySprint_JVmeasurement' selected.
- Data:** Shows a hierarchical view of data annotations for 'JVMeasurement', 'BaseMeasurement', and 'Measurement'. The 'JVMeasurement' panel shows annotations for 'label_quantity: data_file', 'EntryData', 'steps', 'samples', 'instruments', 'results', 'quantity_definitions', 'name', 'datetime', 'lab_id', 'description', 'method', and 'location'. The 'BaseMeasurement' panel shows annotations for 'Measurement', 'steps', 'samples', 'instruments', 'results', 'quantity_definitions', 'name', 'datetime', 'lab_id', 'description', 'method', and 'location'. The 'Measurement' panel shows annotations for 'Activity', 'steps', 'samples', 'instruments', 'results', 'quantity_definitions', 'name', 'datetime', 'lab_id', 'description', 'method', and 'location'.
- Logs:** Shows a search bar and a list of logs.

Showcase in Nomad on experimental plan

Go to Nomad (

https://nomad-hzb-se.de/nomad-oasis/gui/search/solarcells?upload_id=JQr0kd9SQmevZ_BF_JGlcg

)

https://nomad-hzb-se.de/nomad-oasis/gui/user/uploads/upload/id/JQr0kd9SQmevZ_BF_JGlcg/entry/id/iW1BoRMcxzWgEn-VvO4Gjg8aSlbj

Showcase in Nomad on experimental plan

The screenshot displays the Nomad software interface, which is used for managing experimental plans. The interface is divided into several panels:

- Top Navigation:** Includes 'PUBLISH', 'EXPLORE', 'ANALYZE', and 'ABOUT' menus. The user is logged in as 'Michael Götte'.
- Overview Panel (Left):** Contains metadata for the plan, such as 'Number of substrates: 12', 'Substrates per subbatch: 4', and 'ID: HZB_FeUn_20231213_AF-FU-Batch-5'. It also features a rich text editor for the description.
- Step Panel (Middle-Left):** Shows the 'Step' configuration for 'Sam 2PACz spincoating'. It includes 'QUANTITIES' (Name, Vary parameters), 'SUB SECTIONS' (batch_processes), and 'REFERENCES'.
- Table:** A table listing chemical components used in the process.

name	chemical_volume	chemical_mass	concentration_m
(2-(9H-Carbazol-9-yl)ethyl)phosphonic acid 3.0 milligram		3.0 milligram	
Anhydrous ethanol			
- Layer Properties Panel (Right):** Details the 'LayerProperties' for the 'Hole Transport Layer', including the 'layer_material' (P1H14C14N103).
- Main Content Area (Right):** Displays the 'Position in experimental plan' and a detailed 'Description' of the process: 'Blowing off dust with Nitrogen' and 'Drop 100 ul of 2PACz solution on Substrate, Resting for 5-10 s; Start Spin Coating'.

Showcase in Nomad on using jupyter hub

Go to Nomad (

https://nomad-hzb-se.de/nomad-oasis/north/user/michaelgoette/jupyter/lab/tree/uploads/data-analysis-feray-l3tq0kSiQDWS5Uxj91m7rw/jv_analysis_feray.ipynb

)

Showcase in Nomad on using jupyter hub

The screenshot displays a Jupyter Notebook environment with the following content:

Plotting JV data over anti solvent dropping time

```
[1]: import matplotlib.pyplot as plt
from jv_analysis import get_jv_data, plot_quantity_over_jv, get_ids_in_batch, plot_df, get_synthesis
#user = michael.goette@helmholtz-berlin.de
Username michael.goette@helmholtz-berlin.de
.....

[2]: batch_ids = []
# Set Batch ID add other ids for more batches
batch_ids.append("HZB_FeUn_20231213_AF-FU-Batch-5")

quantities = ["layer0/layer_type", "layer0/layer_material_name"]
methods = ["SpinCoating", "SlotDieCoating"]

##### dont change
jv_quantities=["efficiency", "fill_factor", "short_circuit_current_density", "open_circuit_voltage", "cell_name"]
sample_ids = []
for batch_id in batch_ids:
    sample_ids.extend(get_ids_in_batch(batch_id))
samples_of_batches = [(sample_id, get_entryid(sample_id)) for sample_id in sample_ids if get_entryid(sample_id)]
df_j, curve_data = get_jv_data(samples_of_batches, jv_quantities)
df_q = get_synthesis(samples_of_batches, methods, quantities)
#####
```

Data table and Plot of best jv data

```
[3]: # change for different minimum
filter_for_efficiency = 3

# table
df_filtered= df_j[df_j["efficiency"] > filter_for_efficiency].sort values("efficiency", ascending=False)
columns = ["sample_id", "cell_name", "efficiency", "fill_factor", "short_circuit_current_density", "open_circuit_voltage"]
import plotly.graph_objects as go
fig = go.Figure(data=[go.Table(
    header=dict(values=list(columns),
                line_color="darkslategray",
                fill_color="lightskyblue",
                align="left"),
    cells=dict(values=[df_filtered[c] for c in columns], # 2nd column
                line_color="darkslategray",
                #fill_color="lightcyan",
                align="left"), columnwidth=[2.5,1,1,1,1,1])
])
fig.update_layout(height=350, width=1000, margin=dict(l=20, r=20, t=20, b=20))
fig.show()
```

sample_id	cell_name	efficiency	fill_factor	short_circuit_current	open_circuit_voltage
HZB_FeUn_20231213_AF-FU-Batch-5_2_0	d.Reverse_Light	9.76073	0.5808232925	15.78487083	1.06462643
HZB_FeUn_20231213_AF-FU-Batch-5_2_0	d.Forward_Light	9.67227344	0.5758971661000001	15.991075	1.0502821
HZB_FeUn_20231213_AF-FU-Batch-5_2_0	d.Reverse_Light	9.57859583	0.5684614069	15.80342292	1.06622706
HZB_FeUn_20231213_AF-FU-Batch-5_2_0	d.Forward_Light	9.47165625	0.5611945588	16.08838542	1.04905924
HZB_FeUn_20231213_AF-FU-Batch-5_2_0	d.Reverse_Light	9.12235479	0.5391709676	15.8772917	1.06559475
HZB_FeUn_20231213_AF-FU-Batch-5_2_0	d.Forward_Light	9.01665312	0.5319522351	16.24510833	1.04339827
HZB_FeUn_20231213_AF-FU-Batch-5_2_0	f.Reverse_Light	8.97474375	0.5645703078	14.94438125	1.06371696
HZB_FeUn_20231213_AF-FU-Batch-5_2_0	f.Reverse_Light	8.86789021	0.5552485389	14.97894167	1.0662321
HZB_FeUn_20231213_AF-FU-Batch-5_2_0	f.Forward_Light	8.70931094	0.5470829676000001	15.20687917	1.04886458
HZB_FeUn_20231213_AF-FU-Batch-5_2_0	f.Forward_Light	8.6211875	0.5374163688	15.3012625	1.04840464
HZB_FeUn_20231213_AF-FU-Batch-5_2_0	f.Reverse_Light	8.59521833	0.5296169095	15.10764375	1.07423262
HZB_FeUn_20231213_AF-FU-Batch-5_2_1	d.Reverse_Light	8.433295	0.4873905502000000	16.26336042	1.06392227
HZB_FeUn_20231213_AF-FU-Batch-5_2_0	f.Forward_Light	8.32538583	0.5099008457	15.53976667	1.0506889

Would you like to receive official Jupyter news?
 Please read the privacy policy
 [Open privacy policy](#) Yes No

Showcase in Nomad on using jupyter hub

The screenshot displays a Jupyter Notebook environment with the following components:

- Code Cell [4]:** Building a layer stack by combining columns from a DataFrame.
- Code Cell [5]:** Filtering data for minimum efficiency and plotting box plots.
- Box Plot:** A plot showing efficiency for different layer stacks. The y-axis is labeled 'efficiency' and ranges from 3 to 10. The x-axis shows six layer stack categories. A dropdown menu for 'sample_id' is visible.
- Code Cell [10]:** Finding the best curves by identifying the maximum efficiency for each layer stack.

```
[4]: # building the layer stack
##### dont change
import pandas as pd
def combine_columns(row):
    layers = [row[c] for c in row.index if "layer_material_name" in c and not pd.isnull(row[c])]
    return "-".join(layers)
df_q['layer_stack'] = df_q.apply(combine_columns, axis=1)
##### dont change

[5]: ### change filter for minimum efficiency
filter_for_efficiency = 3

##### dont change
df = df_q.merge(df_j, on=["entry_id", "sample_id"])
plot_df(df[df["efficiency"] > filter_for_efficiency], jv_quantities, q_quantities=["layer_stack", "sample_id", "jv_name"])
##### dont change

[10]: # finding best curves
idx_max = df.reset_index().groupby(["layer_stack"])["efficiency"].idxmax()

# plotting
plt.figure(figsize=(14,8))
for index, row in df.iloc[idx_max].iterrows():
    # ... (code partially obscured)
```

Would you like to receive official Jupyter news? Please read the privacy policy. [Open privacy policy](#) Yes No

Showcase in Nomad on using jupyter hub

The screenshot displays a Jupyter Notebook environment with a file browser on the left and a code editor in the center. The code defines a function to find the best curves based on efficiency and then plots them. The plot shows Current Density A/cm² versus Voltage [V] for two different layer stacks: NIOx_Al2O3_MAPbI3 and NIOx_MAPbI3. The legend provides the following parameters for each curve:

Layer Stack	PCE	Voc	Jsc	FF
NIOx_Al2O3_MAPbI3	9.761	1.065	15.785	0.581
NIOx_MAPbI3	5.372	1.012	14.024	0.379

The plot shows two sets of curves: a blue set for the NIOx_Al2O3_MAPbI3 stack and an orange set for the NIOx_MAPbI3 stack. Each set includes a solid line for forward current density and a dashed line for reverse current density. The blue curves show higher efficiency and higher current density at higher voltages compared to the orange curves.

```
[6]: # finding best curves
idx_max = df.reset_index().groupby(['layer_stack'])['efficiency'].idxmax()

# plotting
plt.figure(figsize=(14,8))
for index, row in df.iloc[idx_max].iterrows():
    if row['efficiency'] < filter_for_efficiency:
        continue
    cell = row['cell_name'][0]

    curve_fwd = curve_data.get(f'{index} {cell} Forward Light')
    curve_rev = curve_data.get(f'{index} {cell} Reverse Light')

    p = plt.plot(curve_fwd.get('voltage'), curve_fwd.get('current_density'), label=f'PCE: {round(row['efficiency'],3)}.\
                Voc: {round(row['open_circuit_voltage'],3)}, Jsc: {round(row['short_circuit_current_density'],3)}, FF: {round(row['fill_factor'],3)} {row['layer_stack']}'
    plt.plot(curve_rev.get('voltage'), curve_rev.get('current_density'), '--', color=p[0].get_color())
plt.xlabel('Voltage [V]')
plt.ylabel('Current Density A/cm**2')
plt.legend()
plt.show()
```

Would you like to receive official Jupyter news?
Please read the privacy policy.
[Open privacy policy](#) Yes No