A Hands-on Tutorial on Time Series Imputation with ImputeGAP

Quentin Nater & Mourad Khayati

Hands-on Tutorial @KDD'25, Toronto (Canada)
https://imputegap-tutorials.github.io/KDD-2025

https://imputegap.readthedocs.io
v1.1.1 with support of LLMs

August 4, 2025

UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

Schedule

- Section 1: Introduction and Overview (20 min)
- Section 2: ImputeGAP Features (10 min)
- Section 3: Imputation Pipeline Synthesis (60 min)
- Section 4: Downstream Analysis (20 min)
- Break (15 min)
- Section 5: Imputation Explainability (20 min)
- Section 6: Library Extension (20 min)
- Q&A (20 min)

Presenters

- **Mourad Khayati**
  - Ph.D. from University of Zürich
  - Senior Lecturer at the University of Fribourg (Switzerland)
  - Time series analytics, data cleaning, missing data imputation, Time Series Database Systems
  - KDD 2025 Outstanding Reviewer Award, VLDB 2020 Best Experiments and Analysis Paper Award

- **Quentin Nater**
  - PhD student at the University of Fribourg (Switzerland) since 01/2025
  - Web of Things, missing data imputation, multimodal learning
  - Fantasy writer

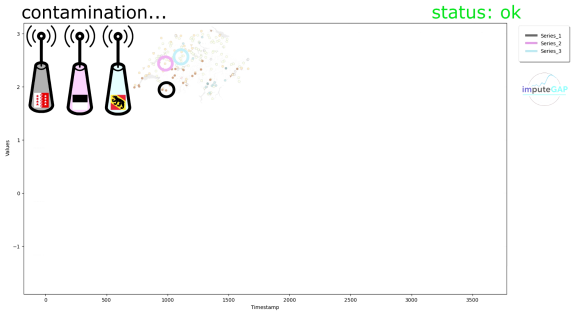
Current Section

Data with Missing Values

- Technical glitches or downtime in data collection systems yield incomplete data.
- Combining data from multiple sources can introduce missing entries.
- Some stocks trade infrequently, so price updates may be missing on certain days.
- During preprocessing, some data might be excluded if it is not worth being retained.
- Data can be withheld due to privacy or proprietary concerns.
- Organizations might not record all the statistical data.
- Participants in a survey do not answer all the questions.

Missing Values in Time Series

- Transmission and hardware problems often occur:
  - Device failure
  - calibration drift
  - Intermittent connectivity
  - External interference
  - Power outages



Missingness in Public Repositories

- Intel-Berkeley Research Lab dataset is missing about 40% (source: A Pipelined Framework for Online Cleaning of Sensor Data Streams, Berkeley technical report, 2005).
- The UCI repository contains datasets with up to 20% missing data (source: A Survey on Missing Data in Machine Learning, J. Big Data, 2021).
- MIMIC Clinical Database has up to 80% missing observations (source: Recurrent Neural Networks for Multivariate Time Series with Missing Values, Scientific Reports, 2018).
- Up to 49% data loss in wearable monitoring (source: Data Quality Evaluation in Wearable Monitoring, Scientific Reports, 2022).

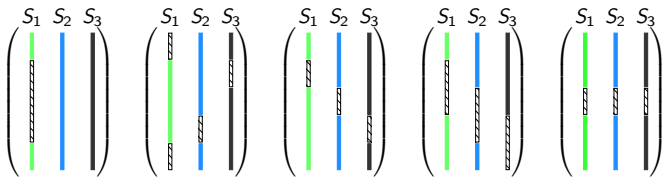
Why Imputation?

- Data management systems and models assume that the data is complete.
- Ignoring missing values has a detrimental impact on time series tasks:
  - ↓ 97% AVG F1 in classification
  - ↓ 55% AVG RMSE in forecasting
- Time Series Database Systems (TSDBs)[1] have begun to incorporate native support for missing value imputation.

[1] khelifati A, Khayati M., Difallah D., Dignös A., and Cudré-Mauroux P.: "TSM-Bench: Benchmarking Time Series Database Systems for Monitoring Applications", PVLDB'23

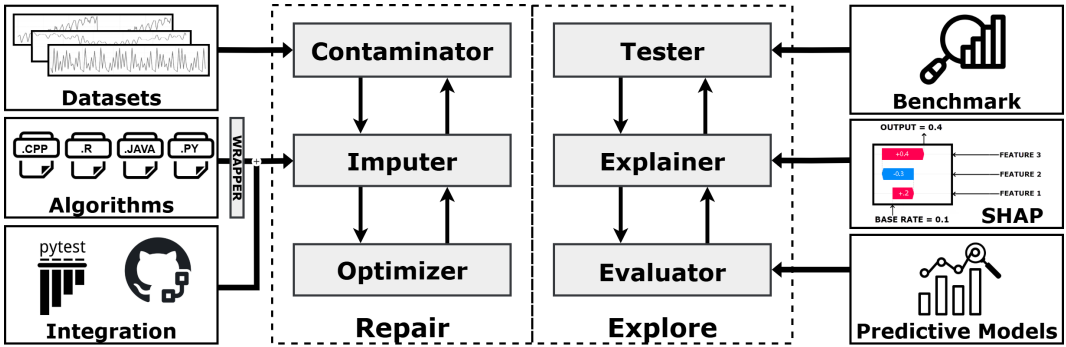
Imputation Caveats

- Compared to tabular data, time series imputation is challenging:
  - temporal dependency
  - high-variability data
  - blocks of consecutive values



Current Section

Python Library

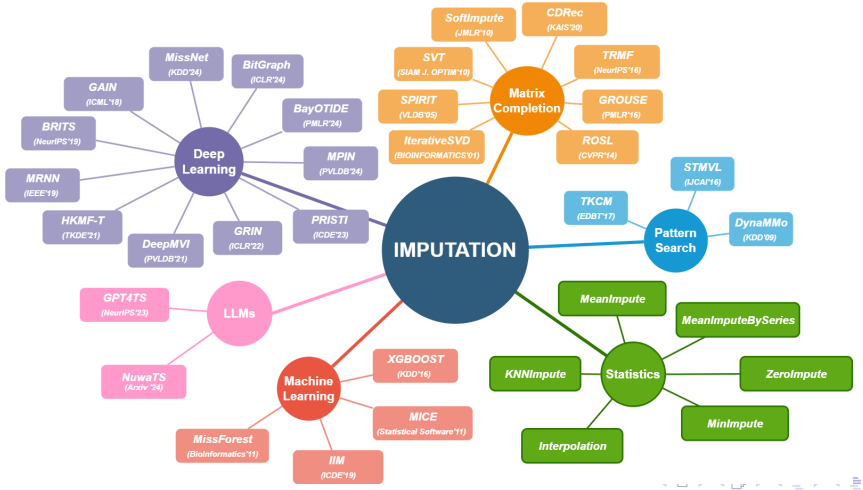


Comparison with Existing Libraries

- Other imputation libraries for time series imputation exist, e.g., PyPOTS, CleanTS, Amelia II, and sktime.
- Limited coverage of imputation algorithms, missingness patterns, and datasets.
- Limited benchmarking customization.
- Do not support the explainability of the results.
- Lack of downstream analysis.
- Lack of integration module.

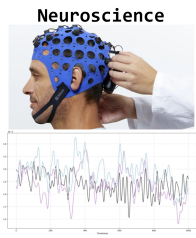
Available Algorithms

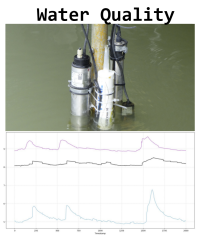
- 34 off-the-shelf imputation algorithms.
- Six families: Matrix Completion, ML, DL, Statistics, Pattern Search, and LLMs.

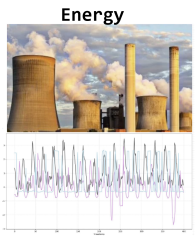


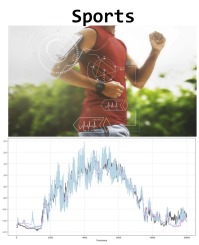
Available Datasets

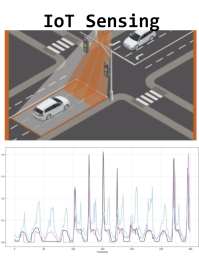
- 17 ready-to-deploy time series datasets.
- 8 real-world applications.

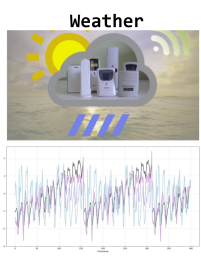


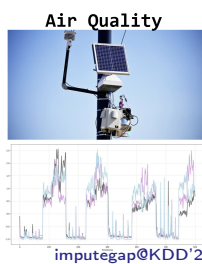






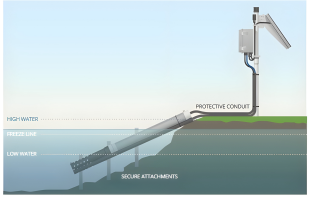


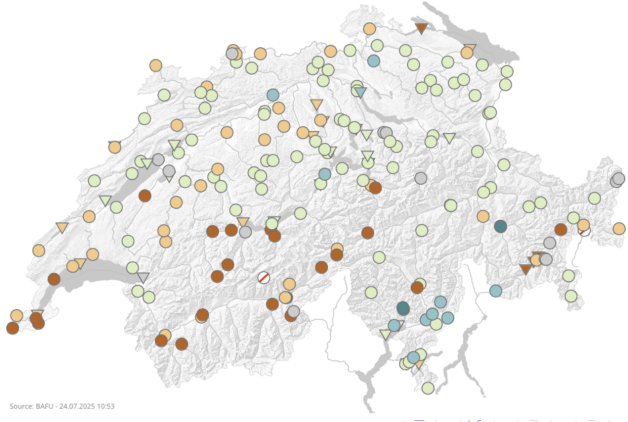




Water Quality

- BAFU[1] deploys sensors to monitor the water quality in Swiss rivers.
- Multivariate time series: discharge, conductivity, oxygen level, temperature, pH values, etc.



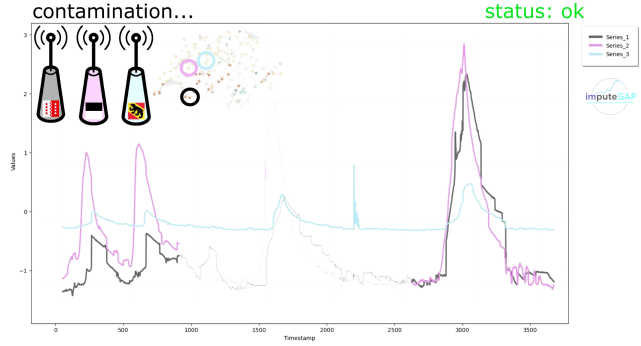


[1] Federal Office for the Environment in Switzerland

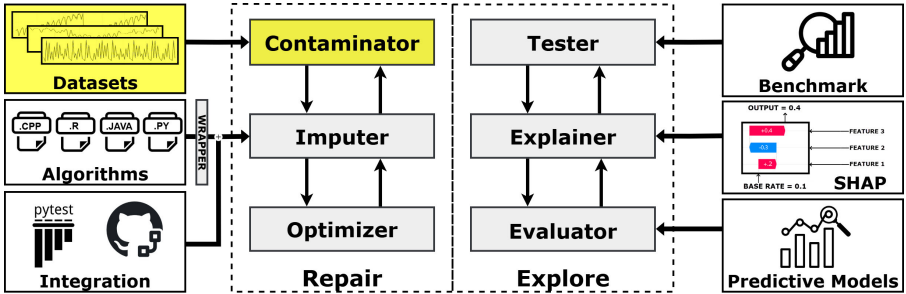
BAFU Dataset

- Monitoring of river levels across three regions, but the sensors suddenly stopped transmitting data.
- The missing data poses safety risks.

Viège (VS) - 21.06.2024

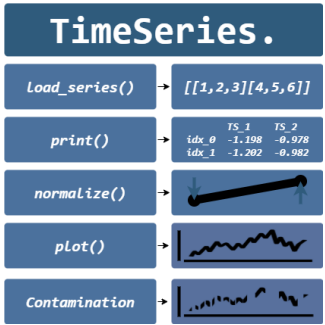


Current Section


Datasets
.CPP
.R
.JAVA
.PY
Algorithms
WRAPPER
pytest
Integration
Contaminator
Imputer
Optimizer
Repair
Tester
Explainer
Evaluator
Explore
Benchmark
OUTPUT = 0.4
+0.4
-0.3
+.2
FEATURE 3
FEATURE 2
FEATURE 1
BASE RATE = 0.1
SHAP
Predictive Models

One Object to Rule Them All

- A single object to manage the entire time series lifecycle, from loading to evaluation.

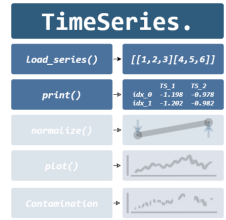


Note

- Input: `file`, `str`, `pathlib.Path`, `list`
- Output: `numpy.ndarray`
- Instance variable : `TimeSeries.data`

```
1 from imputegap.recovery.manager import TimeSeries
2 ts = TimeSeries()
```

Data Loader



- Load time series data internally or from a file.

```
load_series(data, nbr_series=None, nbr_val=None, header=False, replace_nan=False,
    verbose=True) ¶                                                    [source]
```

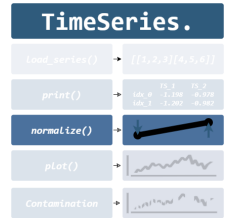
```
1  # load from built-in datasets
2  ts.load_series(utils.search_path("eeg-alcohol"))
3
4  # load your dataset
5  ts.load_series("../imputegap/datasets/kdd.txt"))
6
7  # load as a matrix
8  ts.import_matrix([[1,2,3],[4,5,6]])
9
10 # print your ts
11 ts.print(nbr_series=3, nbr_val=20)
```

Note

Please ensure that your input data satisfies the following format:

- Columns are the series' values
- Column separator: empty space
- Row separator: newline
- Missing values are NaNs

Adaptive Rescaling

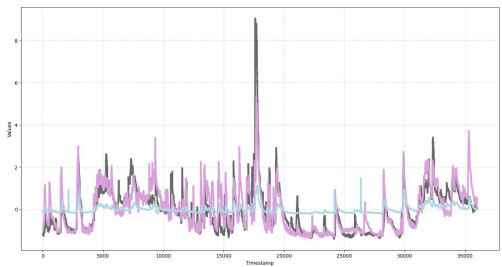


- Normalize the time series data into a different scale.

```
normalize(normalizer='z_score', verbose=True) ¶                    [source]
```

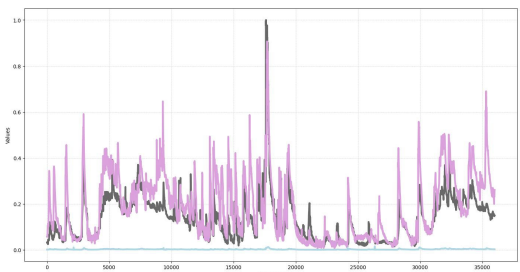
- Z-Score

```
1 ts.normalize(normalizer="z_score")
```

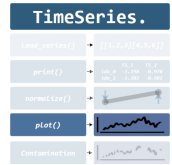


- Min-Max

```
1 ts.normalize(normalizer="min_max")
```



Plotting



- Plot the time series: original, contaminated, and reconstructed.

```
plot(input_data, incomp_data=None, recov_data=None, nbr_series=None, nbr_val=None,
    series_range=None, subplot=False, size=(16, 8), algorithm=None,
    save_path='./imputegap_assets', cont_rate=None, display=True, verbose=True) ¶ [source]
```

Parameters

input_data : numpy.ndarray
The original time series data without contamination.

incomp_data : numpy.ndarray, optional
The contaminated time series data.

recov_data : numpy.ndarray, optional
The imputed time series data.

nbr_series : int, optional
The maximum number of series to plot.

nbr_val : int, optional
The maximum number of values per series to plot.

series_range : int, optional
The index of a specific series to plot. If set, only this series will be plotted.

subplot : bool, optional
Print one time series by subplot or all in the same plot.

size : tuple, optional
Size of the plot in inches. Default is (16, 8).

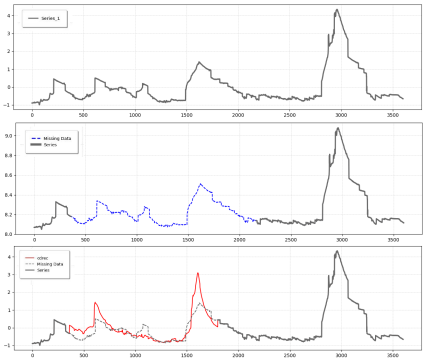
algorithm : str, optional
Name of the algorithm used for imputation.

save_path : str, optional
Path to save the plot locally.

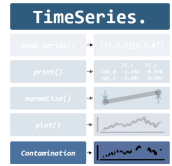
cont_rate : str, optional
Percentage of contamination in each series to plot.

display : bool, optional
Whether to display the plot. Default is True.

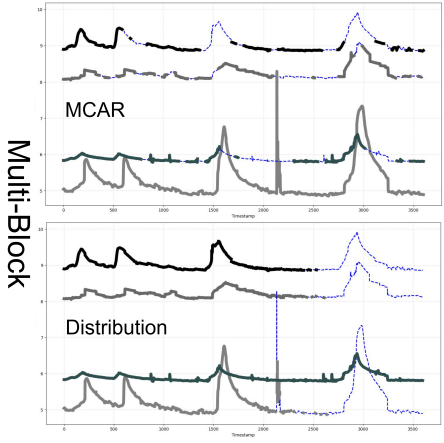
verbose : bool, optional
Whether to display the plot information. Default is True.

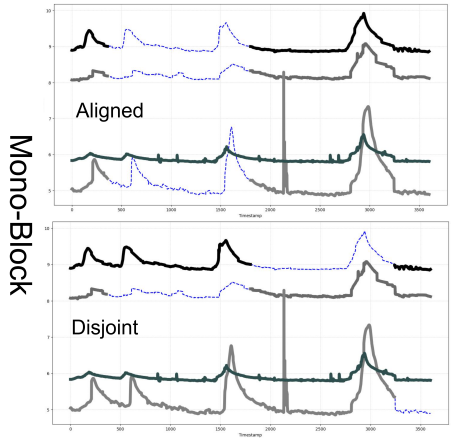


Missingness Patterns

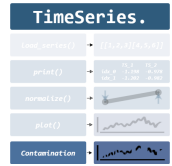


- Simulate two groups of patterns: Multi-block and Mono-block.

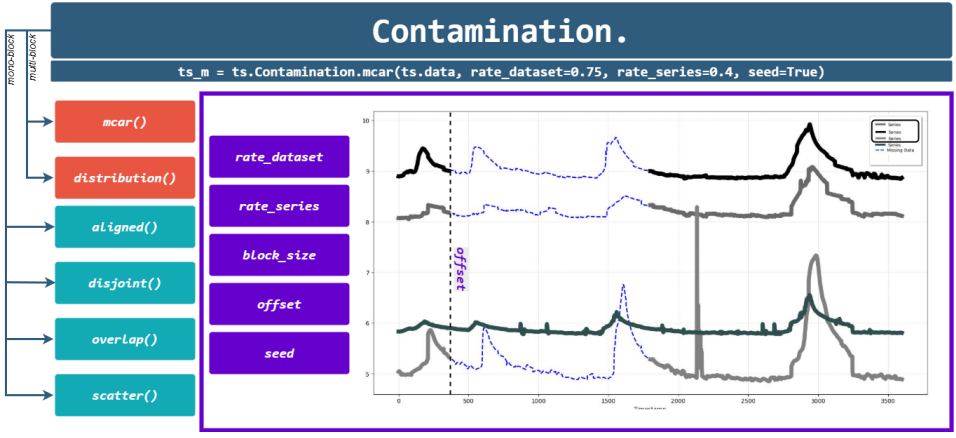




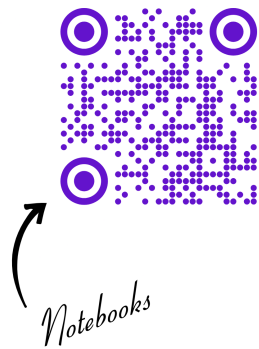
Pattern Customization



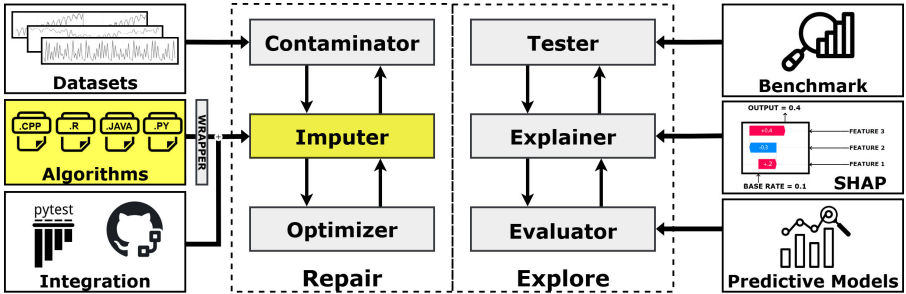
- The patterns can be used with default or customizable configurations.



Demo N°1 - Loading and Contamination

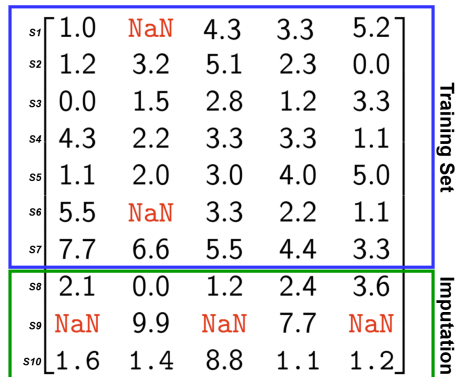


https://imputegap-tutorials.github.io/KDD-2025/html/slides_codes.html


Datasets
.CPP
.R
.JAVA
.PY
Algorithms
WRAPPER
pytest
Integration
Contaminator
Imputer
Optimizer
Repair
Tester
Explainer
Evaluator
Explore
Benchmark
OUTPUT = 0.4
+0.4
-0.3
+.2
FEATURE 3
FEATURE 2
FEATURE 1
BASE RATE = 0.1
SHAP
Predictive Models

Data Splitting Problem

- Using traditional ML splitting, NaN values might appear in the training set.
- The evaluation of Deep Learning techniques might yield data leakage.



Construction of Masks

s1	1.0	NaN	NaN	NaN	5.2
s2	1.2	3.2	5.1	2.3	0.0
s3	0.0	1.5	2.8	1.2	3.3
s4	4.3	2.2	3.3	3.3	1.1
s5	1.1	2.0	3.0	4.0	5.0
s6	5.5	NaN	3.3	2.2	1.1
s7	7.7	6.6	5.5	4.4	3.3
s8	2.1	0.0	1.2	2.4	3.6
s9	NaN	9.9	NaN	7.7	NaN
s10	1.6	1.4	8.8	1.1	1.2

- Control the overall proportion of missing values by artificially inserting missing entries.
- Create a training mask that excludes all missing values with complete time series.
- Create a testing mask for imputation, maintaining a consistent missing ratio across all DL models.

```
original_missing_ratio = get_missing_ratio(cont_data_matrix)
```

```
cont_data_matrix, new_mask = prepare_testing_set(cont_data_matrix, original_missing_ratio,
    block_selection=True, tr_ratio=0.8)
```

Data Leakage Prevention

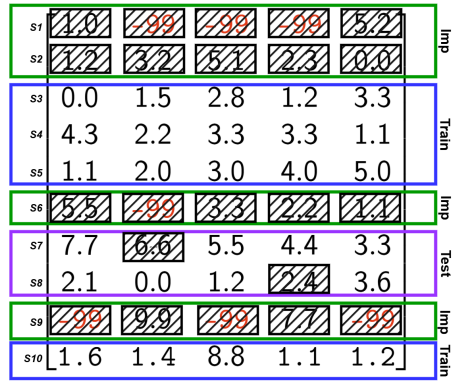
s_1	1.0	-99	-99	-99	5.2
s_2	1.2	3.2	5.1	2.3	0.0
s_3	0.0	1.5	2.8	1.2	3.3
s_4	4.3	2.2	3.3	3.3	1.1
s_5	1.1	2.0	3.0	4.0	5.0
s_6	5.5	-99	3.3	2.2	1.1
s_7	7.7	6.6	5.5	4.4	3.3
s_8	2.1	0.0	1.2	2.4	3.6
s_9	-99	9.9	-99	7.7	-99
s_{10}	1.6	1.4	8.8	1.1	1.2

- Avoid data leakage by replacing all values that must be hidden from the model with large negative constants.
- The model will reveal potential data leakage during evaluation.

```
1 cont_data_matrix = prevent_leakage(cont_data_matrix, new_mask, nan_val=-99)
```

Final Masking

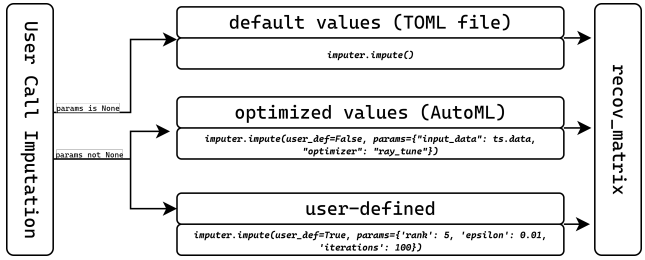
- Split the training set into internal training and testing subsets for the model.
- Artificially mask the internal testing set to evaluate model performance during training.




```
mask_test, mask_valid, nbr_nans = split_mask_bwt_test_valid(cont_data_matrix, test_rate=1, valid_rate=0, nan_val=-99, seed=42)
mask_train = generate_random_mask(cont_data_matrix, mask_test, mask_valid, droprate=0.6, offset=0.1, seed=42)
```

Build Imputation

```
imputer = Imputation.MatrixCompletion.CDRec(ts_m)
imputer.impute()
```




```
def impute(self, user_def=True, params=None):
    if params is not None:
        rank, epsilon, iters = self._check_params(user_def, params)
    else:
        rank, epsilon, iters = utils.load_parameters(query="default", algorithm=self.algorithm)

    self.recov_data = cdrec(incomp_data=self.incomp_data, rank=rank, iters=iters, epsilon=epsilon)

    return self
```

Imputation Evaluation

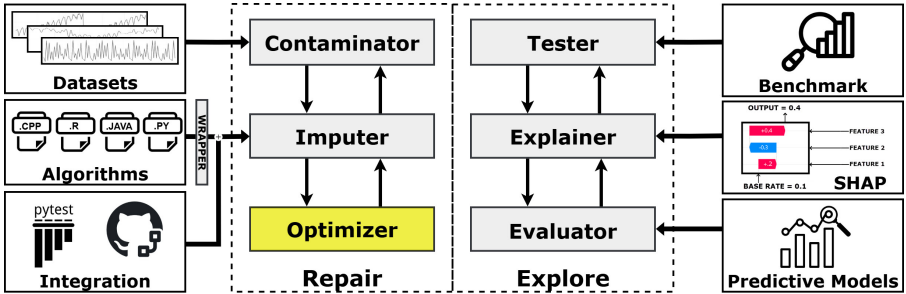
```
class imputegap.recovery.evaluation.Evaluation(input_data, recov_data, incomp_data,
    algorithm='', verbose=True) ¶                                              [source]
```

- Imputation scoring:

```
imputer.score(ts.data, imputer.recov_data)
ts.print_results(imputer.metrics)
```

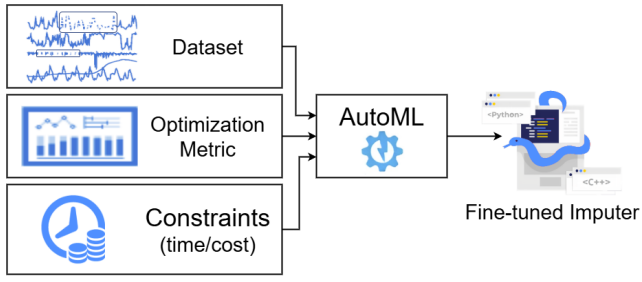
- Dictionary of metrics:

```
recov_vals = self.recov_data[np.isnan(self.incomp_data)]
return {"RMSE": self.compute_rmse(), "MAE": self.compute_mae(), "MI": self.compute_mi(),  "
     CORRELATION": self.compute_correlation()}
```


Datasets
.CPP
.R
.JAVA
.PY
Algorithms
WRAPPER
pytest
Integration
Contaminator
Imputer
Optimizer
Repair
Tester
Explainer
Evaluator
Explore
Benchmark
OUTPUT = 0.4
+0.4
-0.3
+.2
FEATURE 3
FEATURE 2
FEATURE 1
BASE RATE = 0.1
SHAP
Predictive Models

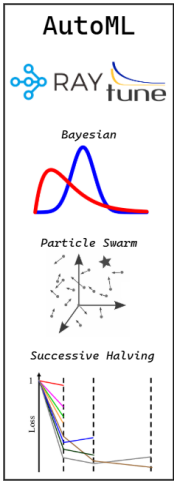
Optimization Methods for Imputation

- Domain-specific imputation algorithms.
- Manually tuning and evaluating each parameter is time-consuming.




```
cdrec_imputer.impute(params={'rank': 5, 'epsilon': 0.01, 'iterations': 100})   # RMSE+0.1
cdrec_imputer.impute(params={'rank': 10, 'epsilon': 0.001, 'iterations': 50}) # RMSE+0.15
cdrec_imputer.impute(params={'rank': 7, 'epsilon': 0.1, 'iterations': 200})    # RMSE-0.1
```

Objective Function

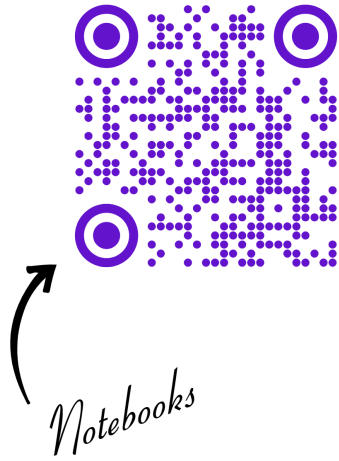


- The objective function is shared across all optimizers.
- Parameter ranges are defined for each algorithm.
- The wider the range, the more computationally expensive the search becomes.

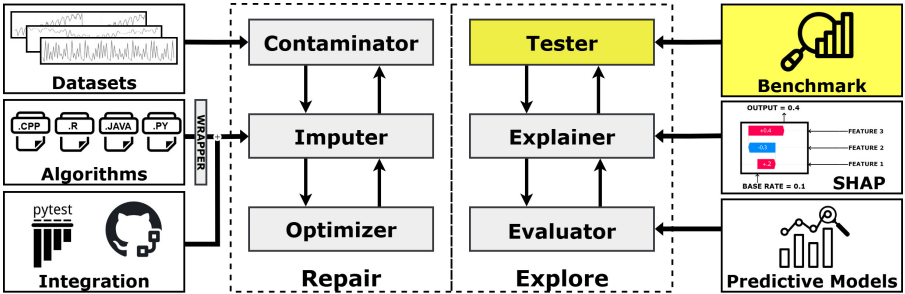
```
RAYTUNE_PARAMS = {
  "cdrec": {
    "rank" : tune.grid_search([i for i in range(2,16,1)]),
    "eps"  : tune.loguniform(1e-6, 1),
    "iters": tune.grid_search([i*50 for i in range(1,4)])
  }
}
```

```
imputer.impute(user_def=False, params={"input_data": ts.data, "optimizer": "ray_tune"})
```

Demo N°2 - Imputation

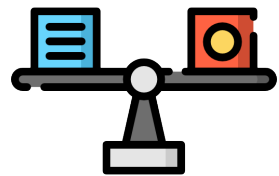


https://imputegap-tutorials.github.io/KDD-2025/html/slides_codes.html


Datasets
.CPP
.R
.JAVA
.PY
Algorithms
pytest
Integration
WRAPPER
Contaminator
Imputer
Optimizer
Repair
Tester
Explainer
Evaluator
Explore
Benchmark
OUTPUT = 0.4
+0.4
-0.3
+.2
FEATURE 3
FEATURE 2
FEATURE 1
BASE RATE = 0.1
SHAP
Predictive Models

Benchmarking

- "Fairness" is key in benchmarking:
  - Are the algorithms properly calibrated?
  - How different are the datasets and metrics?
  - Are the results generalizable?
  - No silver bullet imputation algorithm[1].
- Existing evaluation imputation frameworks:
  - lack of comprehensive end-to-end support
  - limited customization, transparency, and extensibility
  - benchmark failures due to heavy workloads



[1] Khayati M., Lerner A., Tymchenko Z., and Cudré-Mauroux P.: "Mind the Gap: An Experimental Evaluation of Imputation of Missing Values Techniques in Time Series", PVLDB'20 [Best Experiments and Analysis Award]

Workload Execution

18 datasets

7 missingness patterns

34 algorithms

5 optimizers

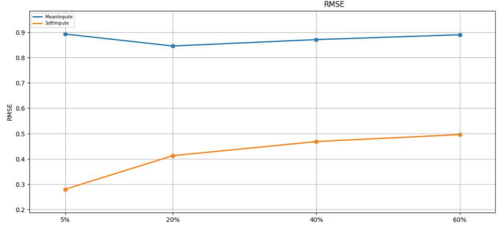
fully customizable missing rate

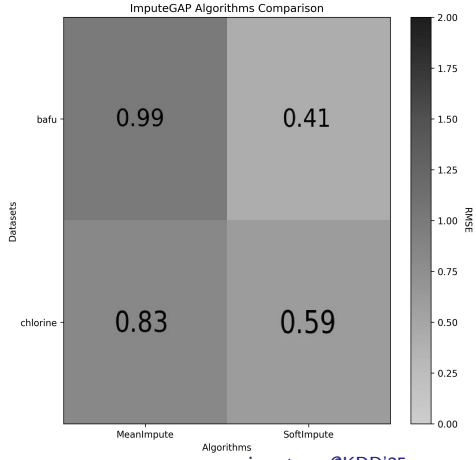
- Over 100k possible combinations.
- On-demand materialization of results.
- GPU accelerators are used when available.
- The optimizer derives optimal parameters based on average runs.

Benchmarking

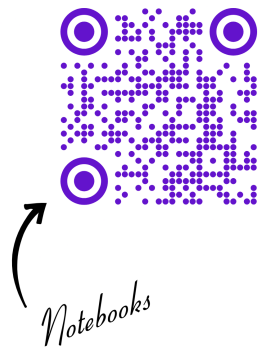
- Fully-customizable evaluation (algorithms, datasets, metrics, patterns, rate, optimizer, etc.).
- Multi-resolution result visualization.

```
bench.eval(datasets=["bafu", "chlorine"],
  patterns=["mcar","aligned"],
  algorithms=["MeanImpute", "SoftImpute"],
  optimizers=["default_params"],
  x_axis=[0.05,0.2,0.4,0.6],
  metrics=["rmse", "runtime"])
```



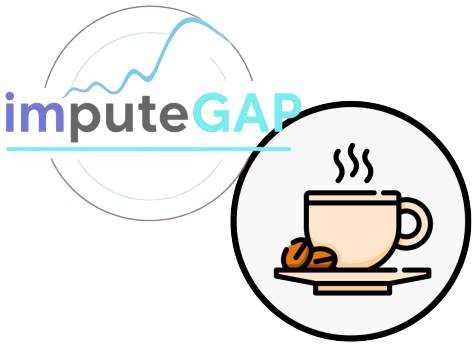


Demo N°3 - Benchmark



https://imputegap-tutorials.github.io/KDD-2025/html/slides_codes.html

Coffee Break

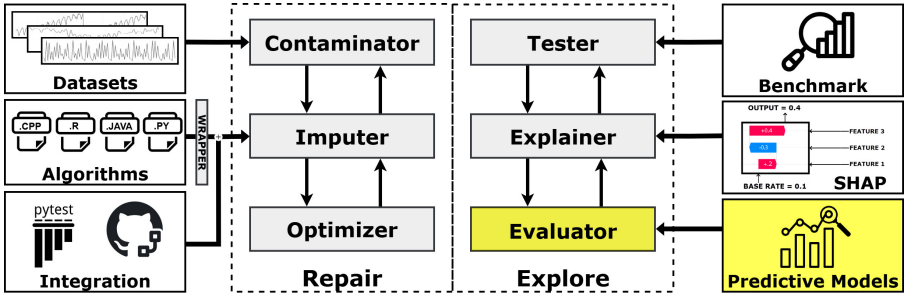


doc!

https://imputegap.readthedocs.io/

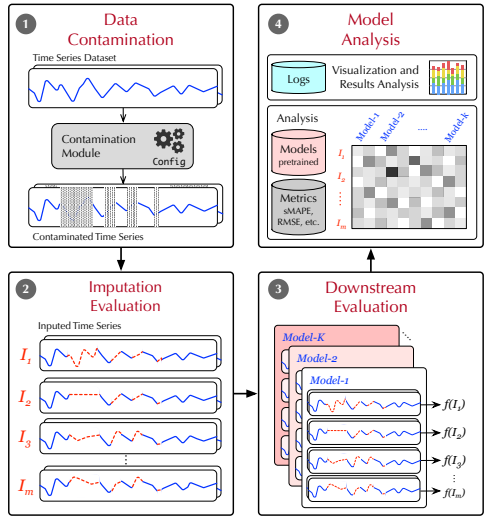
Current Section

Downstream
imputeGAP



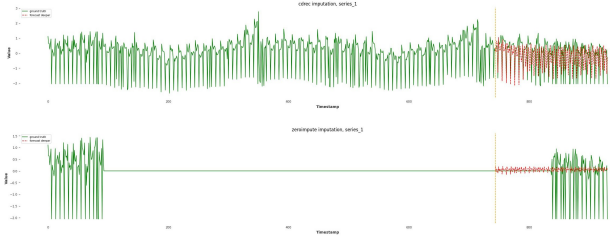
Imputation Impact on Downstream

- Are downstream and upstream gains proportional?
- Do upstream and downstream metrics align?
- Are downstream models comparable when fed with imputed data?
- Do downstream tasks behave differently w/t imputation?



Downstream Implementation

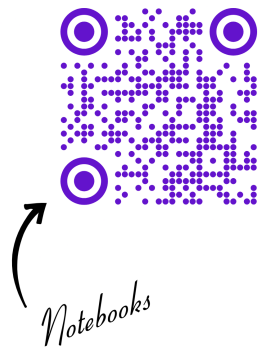
- Current version implements forecasting task (classification in under development)
- 16 parameterizable forecasters from sktime and darts APIs.




```
ImputeGAP downstream models for forcasting : ['arima', 'bats', 'croston', 'deepar', 'ets', 'exp-
    smoothing', 'hw-add', 'lightgbm', 'lstm', 'naive', 'nbeats', 'prophet', 'sf-arima', 'theta', '
    transformer', 'unobs', 'xgboost']
```

```
def downstream_analysis(self):
    task = self.downstream.get("task", "forecast")
    model = self.downstream.get("model", "lightgbm")
    baseline = self.downstream.get("baseline", "zero-impute")
```

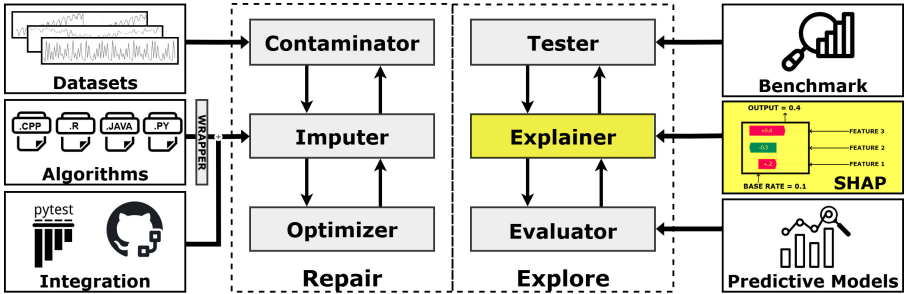
Demo N°4 - Downstream Analysis



https://imputegap-tutorials.github.io/KDD-2025/html/slides_codes.html

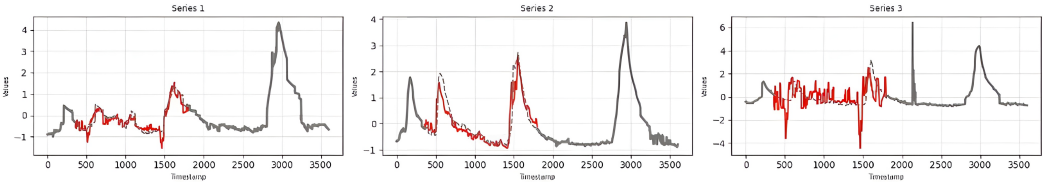
Current Section

Explainer
imputeGAP



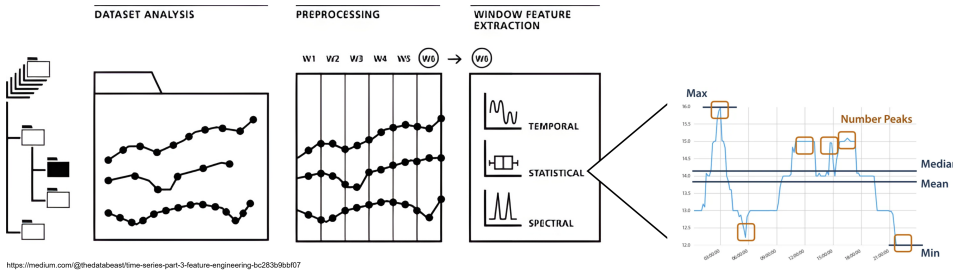
Imputation Fine-grained Analysis

- The evaluation metrics are useful, but can we interpret the imputation results?
- Is the performance of imputation algorithms consistent?
- Which factors contribute to it?



Time Series Characterization

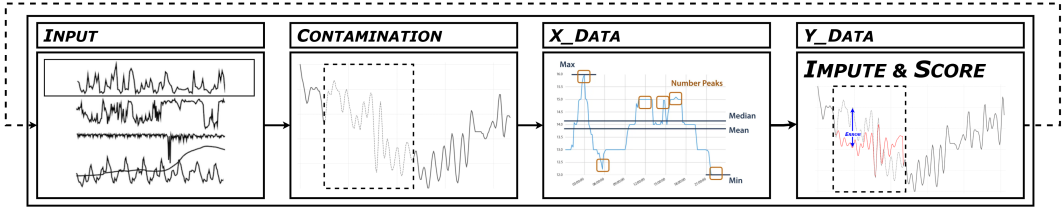
- Interpret imputation results and identify the factors influencing their quality.
- Using feature extractors, we assess imputation quality by analyzing feature-level characteristics of the contaminated data.



https://medium.com/@thedatabeast/time-series-part-3-feature-engineering-bc283b9bbf07
https://www.sciencedirect.com/science/article/pii/S2352711020300017

Explainer Dataset

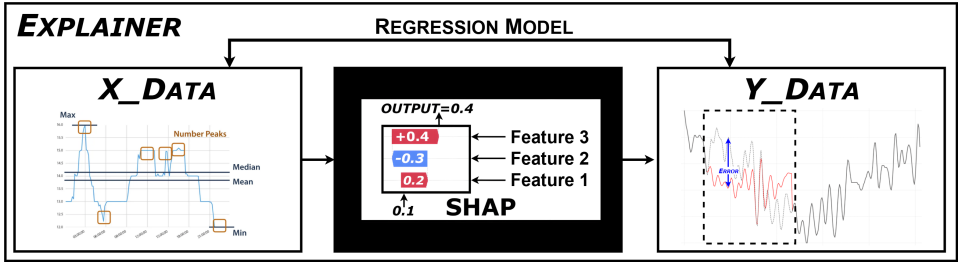
- Iterative process: extract features for each time series contamination.




```
for si in range(0, limit):  # 1) series iterator
    tmp = TimeSeries.import_matrix(data)
    ts_m = utils.config_contamination(tmp, si, pattern="aligned", series_rate=0.6) # 2) contaminate si

    extracted_features = self.extractor(ts_m, categories, features) # 3) extract features

    imputer.impute(user_def=True, params=params)  # 4) impute
    imputer.score(input_data)  # 4) score

    x_data.append(extracted_features)  # store data
    y_data.append(imputer.metrics)  # store labels
```

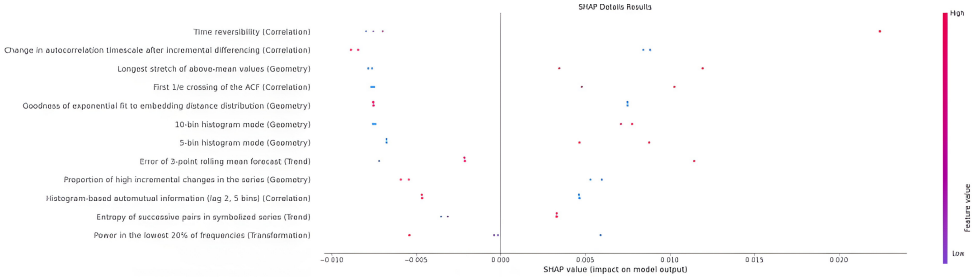
Regression Model

- We use SHAP to learn the relationship between feature values and the imputation score.
- We compute a weight that reflects the contribution to the input.



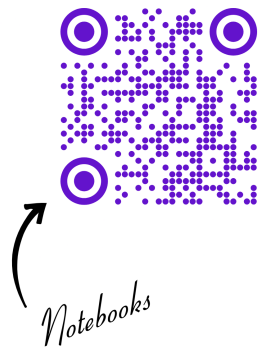

```
model = RandomForestRegressor()
model.fit(x_train, y_train)
exp = shap.KernelExplainer(model.predict, x_test)
shval = exp.shap_values(x_test)
```

Feature Contribution




```
1 Feat:5   CDRec/Bafu  score=13.21  Correlation  Time reversibility
2 Feat:12  CDRec/Bafu  score=10.17  Correlation  Change of autocorrelation timescale after incr. diff
3 Feat:7   CDRec/Bafu  score=9.07   Geometry     Longest stretch of above-mean values
```

Demo N°5 - Explainer



https://imputegap-tutorials.github.io/KDD-2025/html/slides_codes.html

Current Section

Integration

- ImputeGAP is fully extensible and customizable
- We provide a C++ wrapper that can be easily extended to other languages such as JAVA, MATLAB, or R.

add your dataset

add your missingness pattern

add your algorithm

add your optimizer

Integration Python

- Add your custom algorithm rationale.

your_llm.py

```python
def your_llm(ts_m, par_1, par_2, tr_ratio, logs, verbose):
    start_time = time.time()
    recov_data = your_algo(ts_m, par_1, par_2, tr_ratio, logs, verbose)
    end_time = time.time()
    return recov_data
```

- Assign it to the appropriate algorithm family by creating a dedicated class.

imputation.py

```python
class Imputation:
    class LLMs:
        class YourLLM(BaseImputer):
        algorithm = "your_llm"
        def impute(self, user_def=True, params=None, tr_ratio=0.9):
            from imputegap.algorithms.your_llm import your_llm

            self.recov_data = your_llm(ts_m=self.incomp_data, par_1, par_2, tr_ratio, self.logs,
                  self.verbose)

            return self
```

Integration Toolbox

- Define default values the hyperparameters.

utils.py

```python
config = toml.load(filepath)
if algorithm == "your_llm":
    par_1 = int(config[algorithm]['par_1'])
    par_2 = str(config[algorithm]['par_2'])
    return (par_1, par_2)
```

default_values.toml

```toml
[your_algo]
par_1 = 42
par_2 = "val"
```

- Register the algorithm in the toolkit interface for easier access.

utils.py

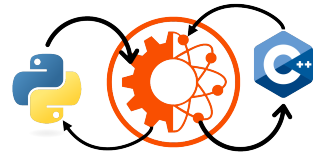
```python
alg = algorithm.lower().replace('_', '').replace('-', '')

if alg == "your_algo":
        imputer = Imputation.YourClass.YourAlgo(incomp_data)

return imputer
```

C++ Wrapper

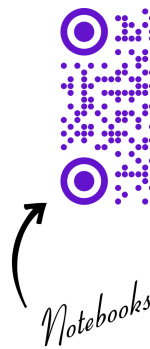
- The wrapper converts input and output variables between Python and C++.
- C++ code is compiled into shared objects (.so files)

```
shared_lib = utils.load_share_lib("lib_cdrec.so")
__ctype_matrix = utils.__marshal_as_native_column(__py_matrix);
shared_lib.cdrec_imputation(__ctype_matrix);

py_matrix = utils.__marshal_as_numpy_column(__ctype_matrix);
```

```
void
cdrec_imputation(double *matrixNative)
{
    arma::mat input = marshal_as_arma(matrixNative, dimN, dimM);
    Algorithms::CDMissingValueRecovery rmv(input);
}
```



Demo N°6 - Integration



https://imputegap-tutorials.github.io/KDD-2025/html/slides_codes.html

A Hands-on Tutorial on Time Series Imputation with ImputeGAP

Thank you!

quentin.nater@unifr.ch
mourad.khayati@unifr.ch

Questions?

https://imputegap.readthedocs.io

References I

[1] Quentin Nater, Mourad Khayati, and Jacques Pasquier
ImputeGAP: A Comprehensive Library for Time Series Imputation.
arXiv, 2025

[2] Mourad Khayati, Guillaume Chaucun, Zakhar Tymchenko, and Philippe Cudré-Mauroux
A-DARTS: Stable Model Selection for Data Repair in Time Series.
Proc. of the 41st IEEE International Conference on Data Engineering (ICDE 2025), Hong Kong.

[3] Mourad Khayati, Quentin Nater, and Jacques Pasquier
ImputeVIS: An Interactive Evaluator to Benchmark Imputation Techniques for Time Series Data.
Proc. of VLDB Endowment (PVLDB 2024), Guangzhou, China.

[4] Mourad Khayati, Alberto Lerner, Zakhar Tymchenko, and Philippe Cudré-Mauroux
Mind the Gap: An Experimental Evaluation of Imputation of Missing Values Techniques in Time Series.
Proc. of VLDB Endowment (PVLDB 2020), Japan.

References II

[5] Luca Althaus, Mourad Khayati, Abdelouahab Khelifati, Anton Dignös, Djellel Eddine Difallah, and Philippe Cudré-Mauroux
SEER: An End-to-End Toolkit for Benchmarking Time Series Database Systems in Monitoring Applications.
Proc. of VLDB Endowment (PVLDB 2024), Guangzhou, China.

[6] Abdelouahab Khelifati, Mourad Khayati, Anton Dignös, Djellel Difallah and Philippe Cudré-Mauroux,
TSM-Bench Benchmarking Time Series Database Systems for Monitoring Applications.
Proc. of VLDB Endowment (PVLDB 2023), Vancouver, Canada.

[7] Mourad Khayati, Ines Arous, Zakhar Tymchenko, and Philippe Cudré-Mauroux,
ORBITS: Online Recovery of Missing Values in Multiple Time Series.
Proc. of VLDB Endowment (PVLDB 2021), Copenhagen, Denmark.