Calm-Data Generator: A Flexible Framework for Synthetic Dataset Creation Under Concept Drift

UNIVERSIDAD
DE MURCIA

ANTONIO GUILLÉN TERUEL

Topics

- Background: Concept Drift, Data generation

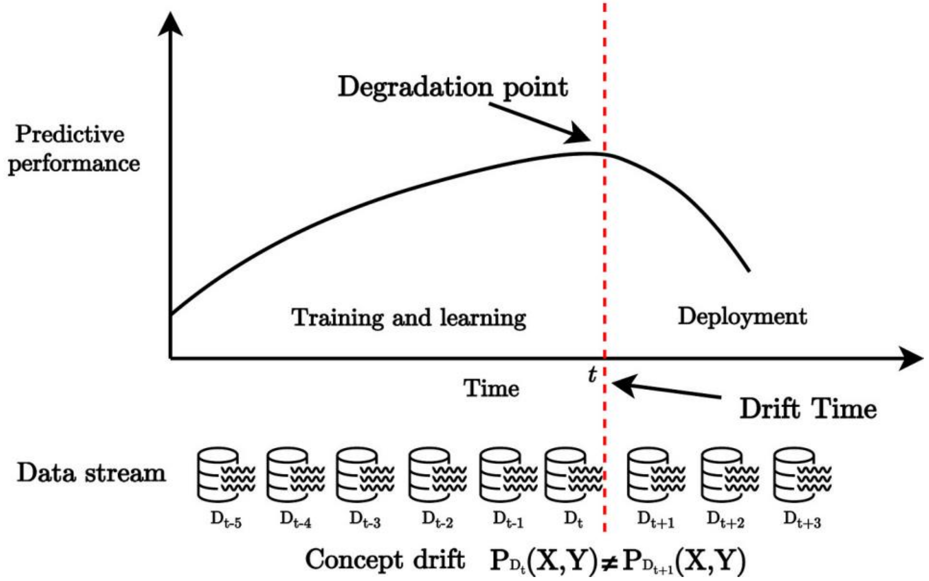
- Methodology: Architecture and Design
- Results
  - Real data simulator
  - Drift injector
  - Synthetic Clinical data: Genes and Proteins
- To do

Concept Drift

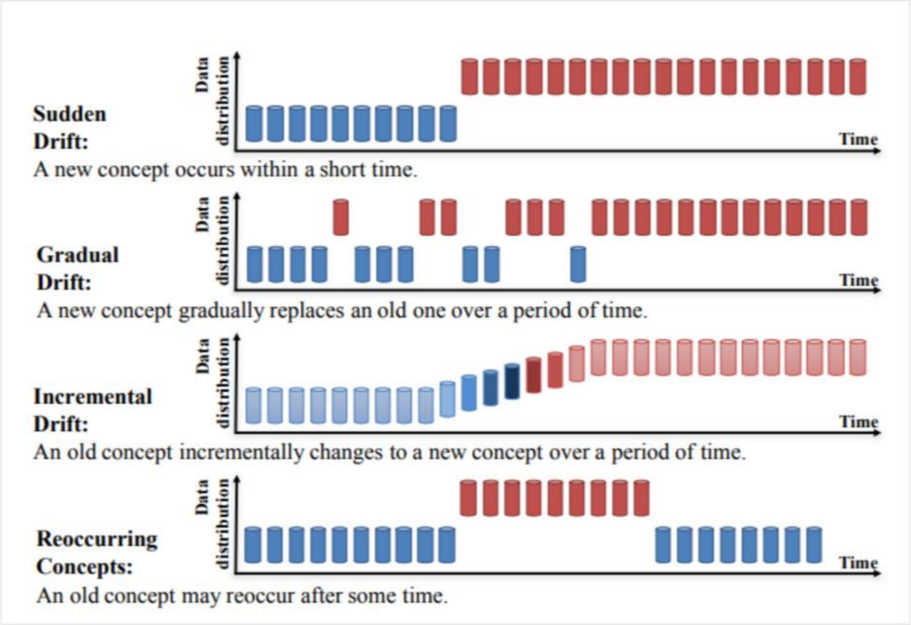
- Change in the distribution of our data over time.

- Very common problem when a model goes into production.

- Problem: Identify and deal with it



Concept drift $P_{D_t}(X,Y) \neq P_{D_{t+1}}(X,Y)$



Sudden Drift:
A new concept occurs within a short time.

Gradual Drift:
A new concept gradually replaces an old one over a period of time.

Incremental Drift:
An old concept incrementally changes to a new concept over a period of time.

Reoccurring Concepts:
An old concept may reoccur after some time.

Data generation: Why simulate data?

- **Real data is hard to share**

- **You rarely get "ground truth" drift**

- **Fair benchmarking needs control**: to compare drift detectors/adaptation methods, you need datasets where you can control

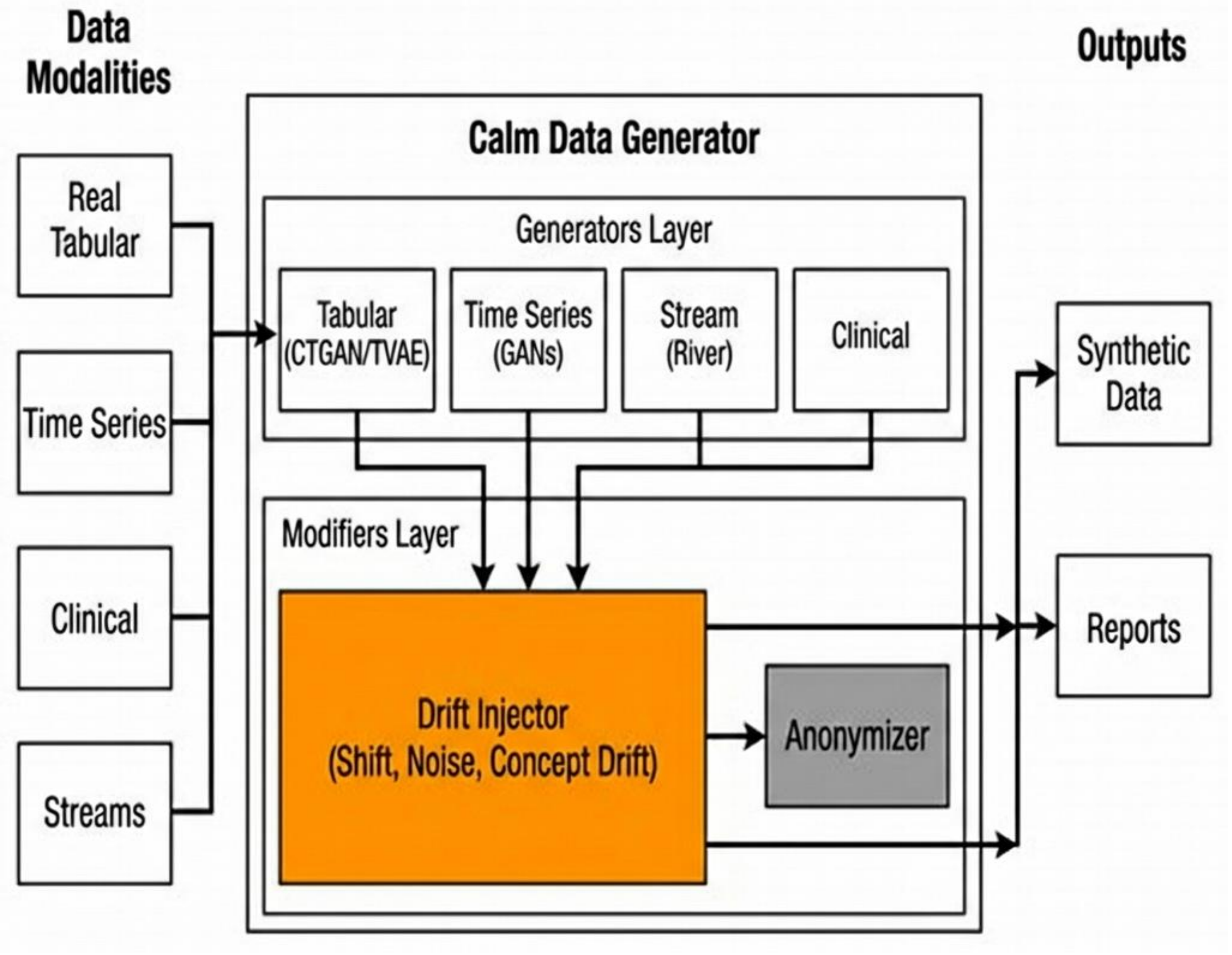
- **Reproducibility:** synthetic generators allow exact re-runs, ablations, and sensitivity analyses

- **Stress-testing models:** simulate rare events, extreme imbalance, or feature shifts that are underrepresented in collected data

- **Domain realism:** generate biomedical-like tabular data (e.g., genes/proteins/clinical variables) while keeping interpretability and drift injection feasible.

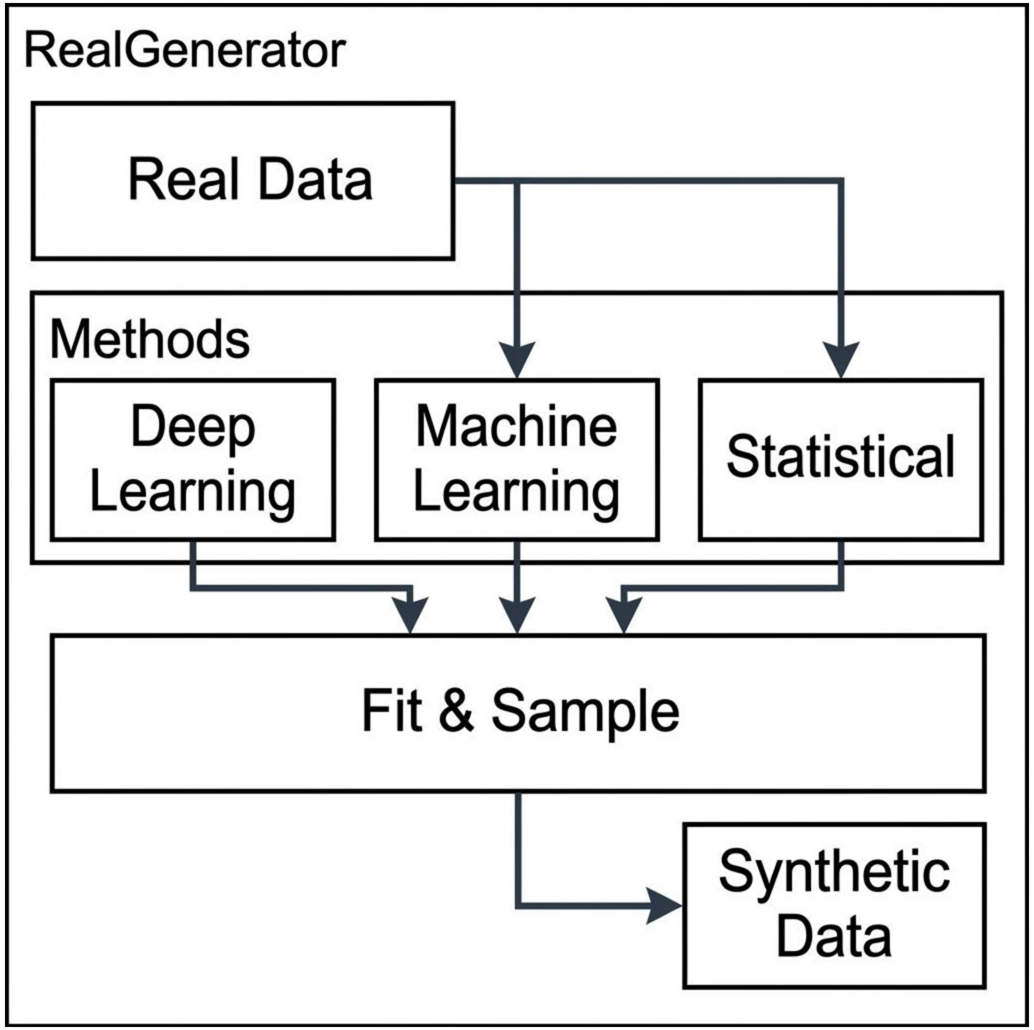
Methodology: Architecture overview

- Unified framework to generate synthetic data from different modalities
- Two main layers: Generators and modifier layers
- Outputs: synthetic datasets + reports



Methodology: Real tabular data generator

- Input: Real tabular data previously curated
- Several simulators:
  - TVAE, CTGAN, …: Deep Learning Methods *(sdv library)*
  - CART (Decision trees), Random Forest: ML Methods
- Fit on real data → Sample strategy
- Synthetic data: Report



Methodology: Stream and time-series data generator

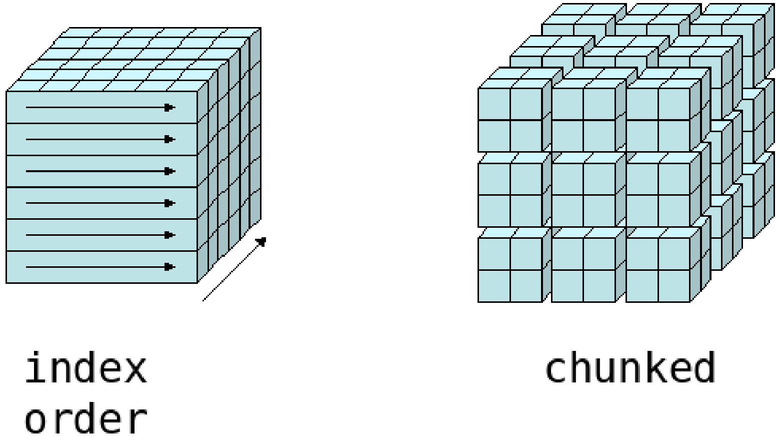
- Input: None

- Several simulators:
  - *River:* Python library to generate data.
  - Some examples of generators: Agrawal, Hyperplane…

- One method to 'chunk' the data into several parts.

- Synthetic data: Report

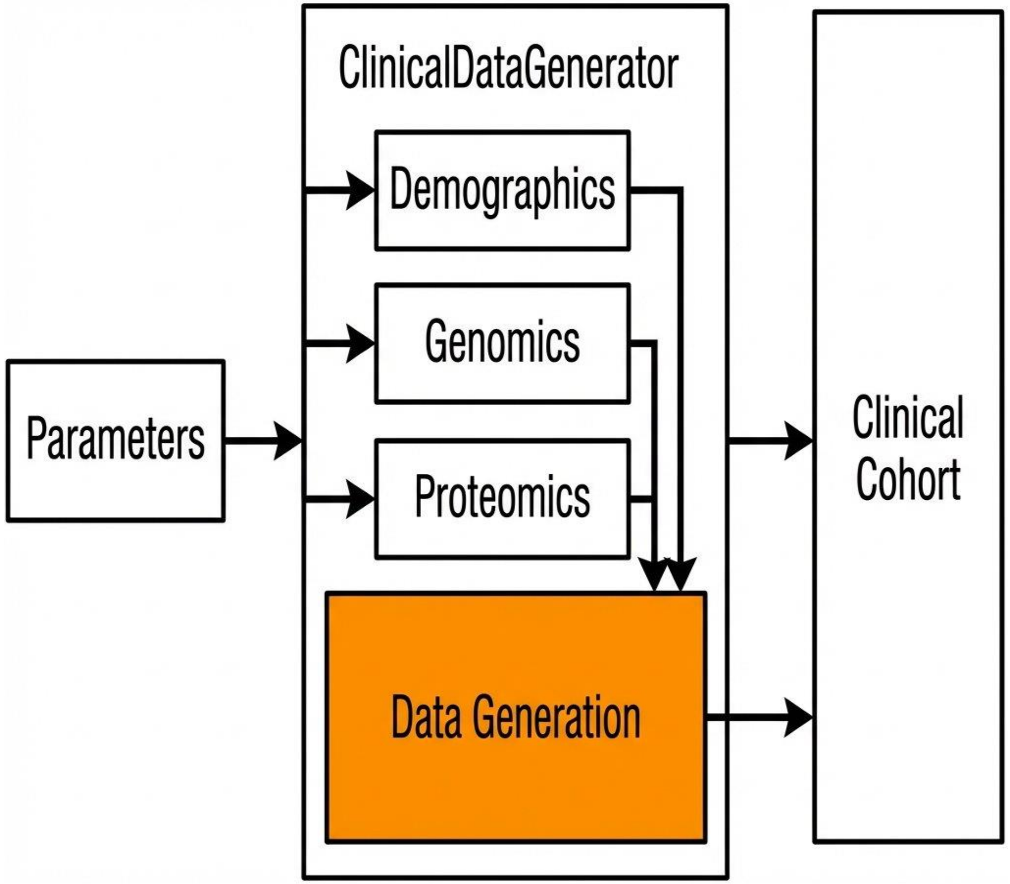
River



Methodology: Clinic data generator – Genes+Proteins

- Specific data generator for proteins,genes and demographic information of a set of patients

- Can be used for testing models before using them in real high-dimensional data



Methodology: Drift injector

- Can be utilised while creating synthetic data.

- Time-related drifts: Abrupt, gradual, incremental, recurrent.

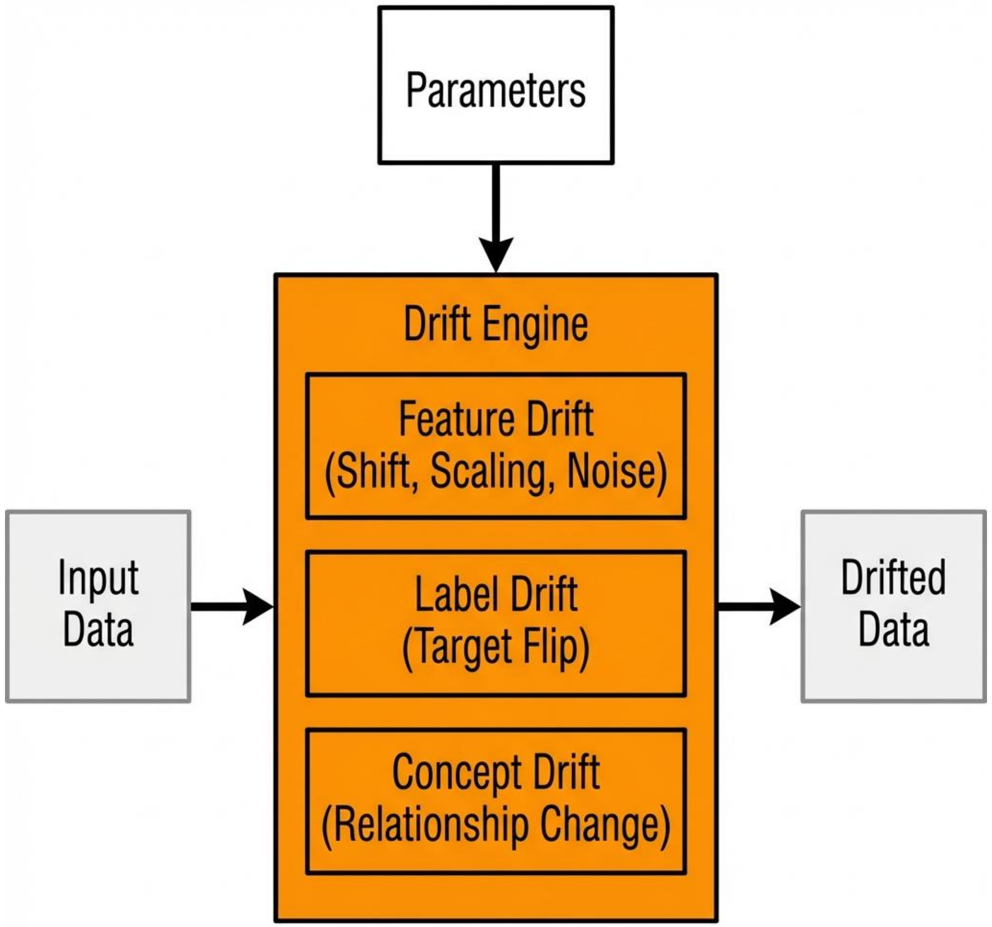
- You can change data from:

    - One specific timepoint
    - One specific chunk
    - One specific index (row)

- Drifts according to:

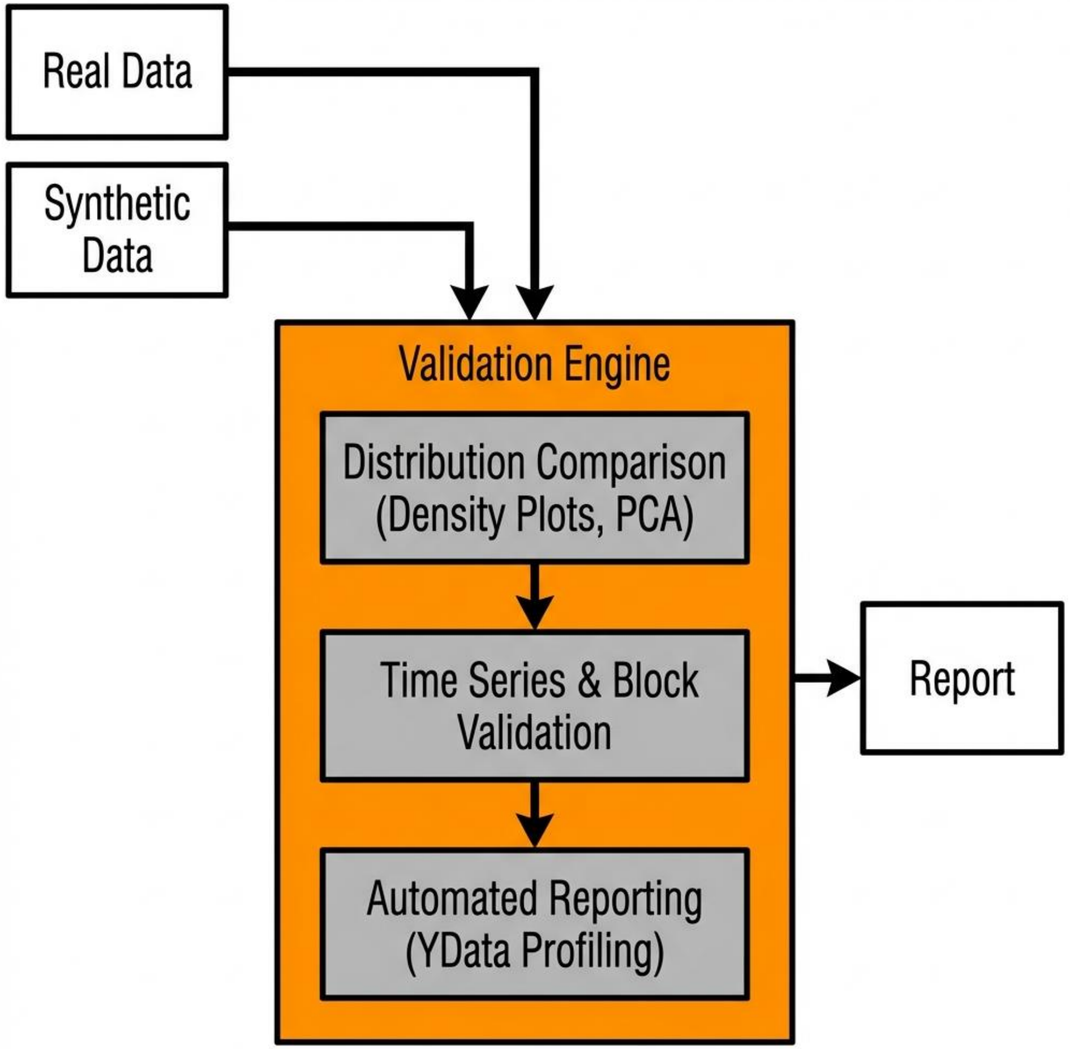
    - Some specific feature or set of features.
    - Some specific conditions of a set of features
    - The target variable (label drift)

- Reports: drifted dataset vs the original dataset



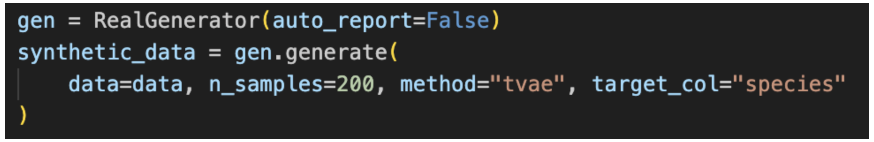
Methodology: Reports of the results

- Plots and statistics to visualise:
  - Real data vs Synthetic data
  - Data with Drift vs Data without Drift
- Comparison between two datasets using:
  - **SDV indicator**: Comparison between distributions
  - **Weighted-SDV indicator**: Penalising by duplicated rows
- Report in a **html** with all the information


Real Data
Synthetic Data
Validation Engine
Distribution Comparison (Density Plots, PCA)
Time Series & Block Validation
Automated Reporting (YData Profiling)
Report

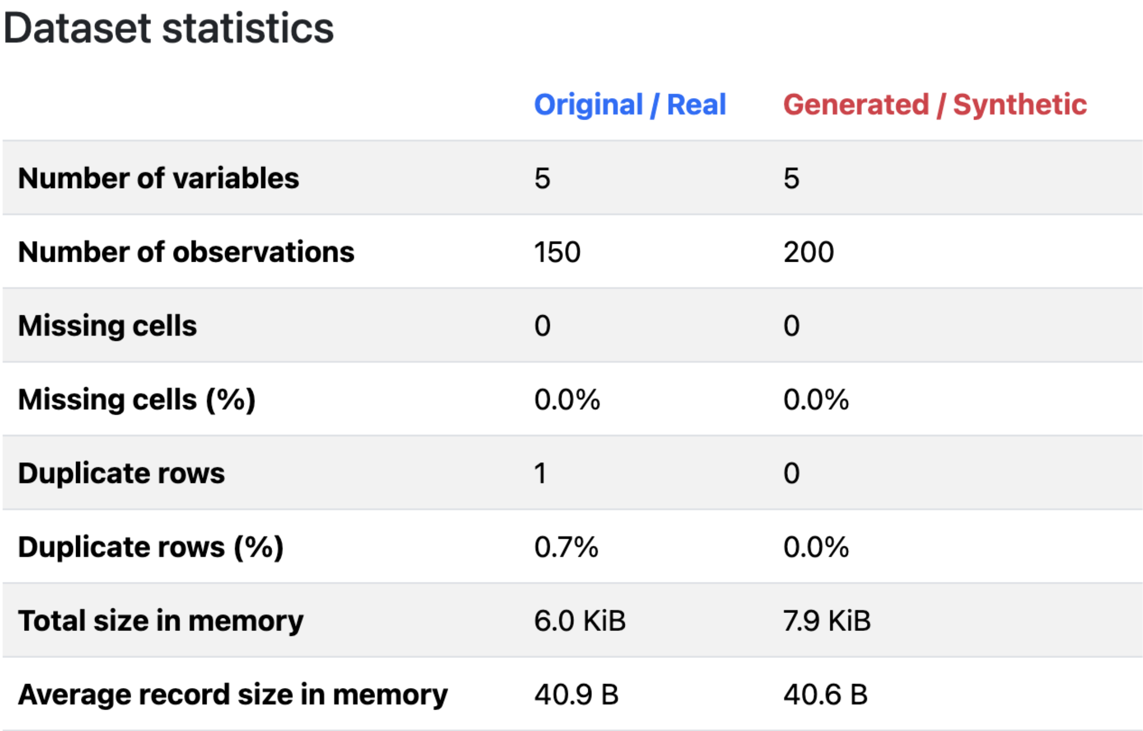
Results: Real data generator – IRIS dataset

- Set-up parameters:
  - Iris dataset as df input
  - Deep Learning method: TVAE
  - 200 rows simulated



```
gen = RealGenerator(auto_report=False)
synthetic_data = gen.generate(
    data=data, n_samples=200, method="tvae", target_col="species"
)
```

- We can run a report of the comparison between real and synthetic dataset

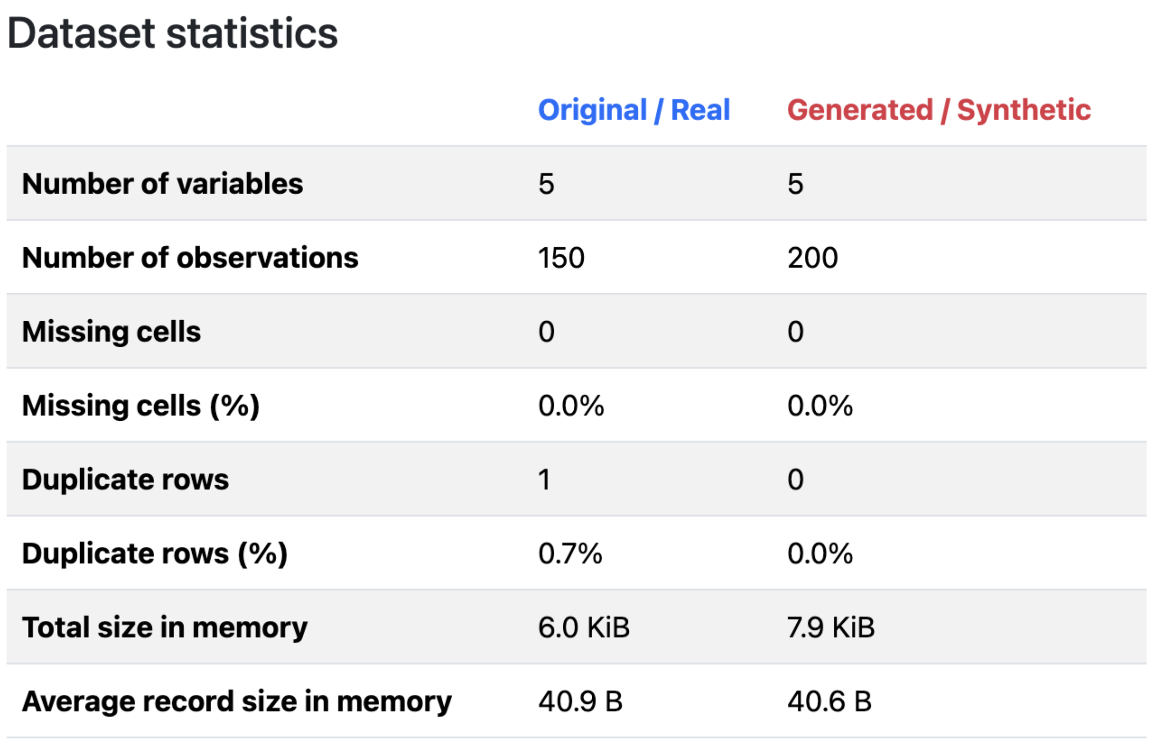


Dataset statistics

	Original / Real	Generated / Synthetic
Number of variables	5	5
Number of observations	150	200
Missing cells	0	0
Missing cells (%)	0.0%	0.0%
Duplicate rows	1	0
Duplicate rows (%)	0.7%	0.0%
Total size in memory	6.0 KiB	7.9 KiB
Average record size in memory	40.9 B	40.6 B

Results: Real data generator – IRIS dataset

- Set-up parameters:
  - Iris dataset as df input
  - Deep Learning method: TVAE
  - 200 rows simulated
- We can run a report of the comparison between real and synthetic dataset

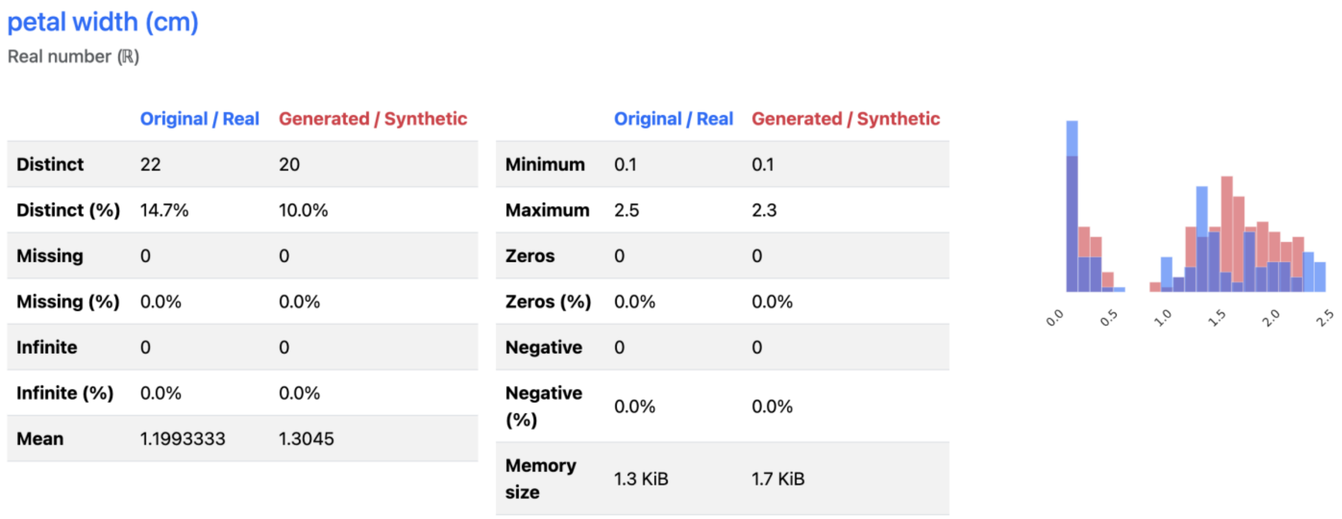


Dataset statistics

	Original / Real	Generated / Synthetic
Number of variables	5	5
Number of observations	150	200
Missing cells	0	0
Missing cells (%)	0.0%	0.0%
Duplicate rows	1	0
Duplicate rows (%)	0.7%	0.0%
Total size in memory	6.0 KiB	7.9 KiB
Average record size in memory	40.9 B	40.6 B

Results: Real data generator – IRIS dataset

- Set-up parameters:
  - Iris dataset as df input
  - Deep Learning method: TVAE
  - 200 rows simulated
- We can run a report of the comparison between real and synthetic dataset



petal width (cm)

Real number (ℝ)

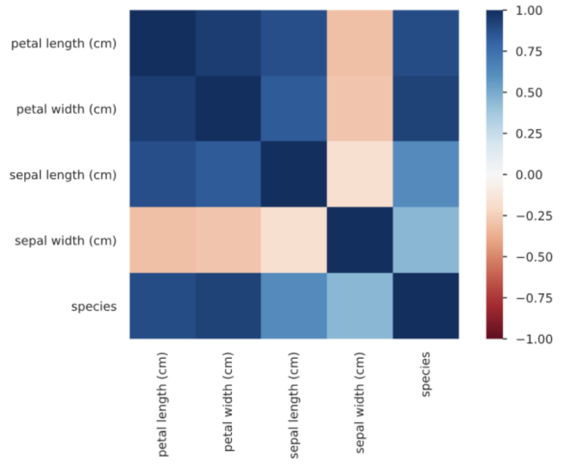
	Original / Real	Generated / Synthetic
Distinct	22	20
Distinct (%)	14.7%	10.0%
Missing	0	0
Missing (%)	0.0%	0.0%
Infinite	0	0
Infinite (%)	0.0%	0.0%
Mean	1.1993333	1.3045

	Original / Real	Generated / Synthetic
Minimum	0.1	0.1
Maximum	2.5	2.3
Zeros	0	0
Zeros (%)	0.0%	0.0%
Negative	0	0
Negative (%)	0.0%	0.0%
Memory size	1.3 KiB	1.7 KiB

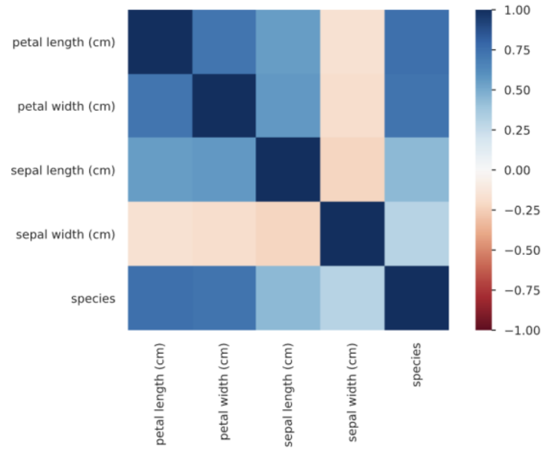
Results: Real data generator – IRIS dataset

- Set-up parameters:
  - Iris dataset as df input
  - Deep Learning method: TVAE
  - 200 rows simulated
- We can run a report of the comparison between real and synthetic dataset

Original / Real



Generated / Synthetic



OVERALL QUALITY

85.6%

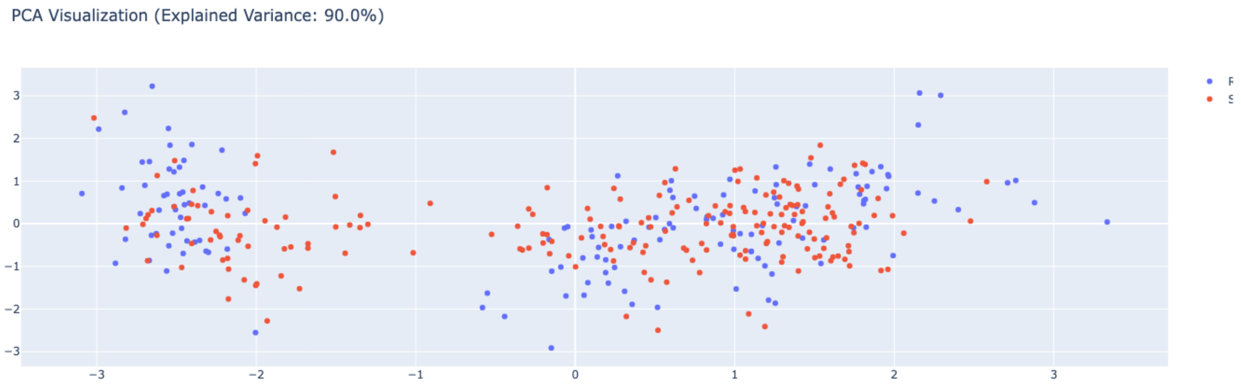
WEIGHTED QUALITY

85.2%

Results: Real data generator – IRIS dataset

- Set-up parameters:
    - Iris dataset as df input
    - Deep Learning method: TVAE
    - 200 rows simulated

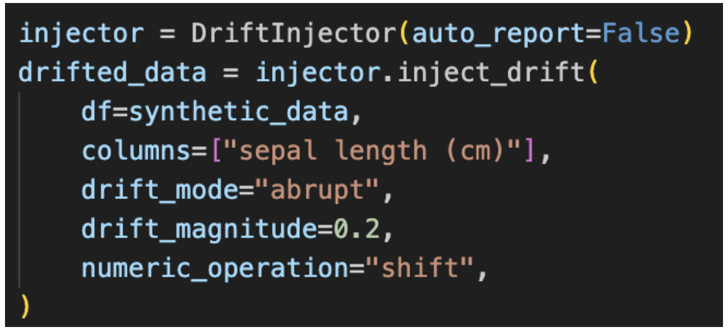
- We can run a report of the comparison between real and synthetic dataset


PCA Visualization (Explained Variance: 90.0%)

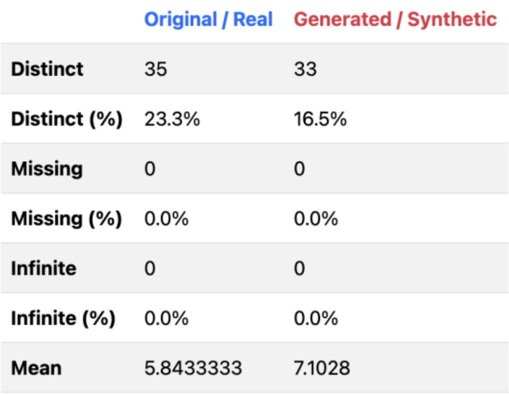
Results: Drift injector – IRIS dataset

- Set-up parameters:
  - Synthetic Iris dataset as df input
  - Drift on a specific column
  - Abrupt drift
  - New values = x+ (mean*magnitude)

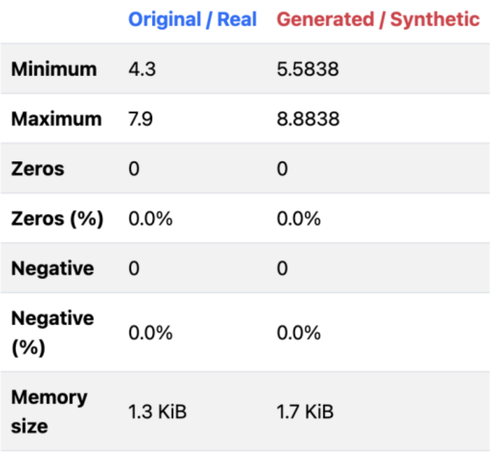
- We can run a report of the comparison between real dataset vs the drifted dataset



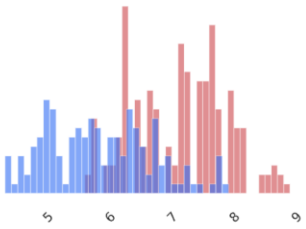
```
injector = DriftInjector(auto_report=False)
drifted_data = injector.inject_drift(
    df=synthetic_data,
    columns=["sepal length (cm)"],
    drift_mode="abrupt",
    drift_magnitude=0.2,
    numeric_operation="shift",
)
```



	Original / Real	Generated / Synthetic
Distinct	35	33
Distinct (%)	23.3%	16.5%
Missing	0	0
Missing (%)	0.0%	0.0%
Infinite	0	0
Infinite (%)	0.0%	0.0%
Mean	5.8433333	7.1028

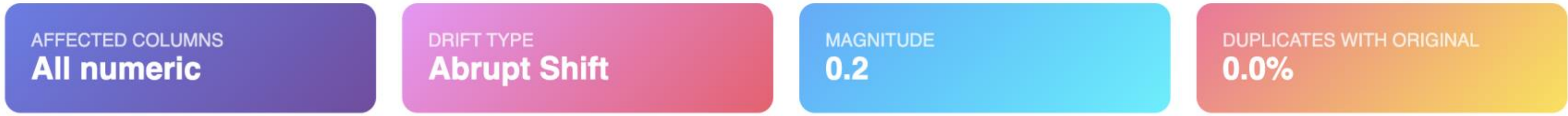


	Original / Real	Generated / Synthetic
Minimum	4.3	5.5838
Maximum	7.9	8.8838
Zeros	0	0
Zeros (%)	0.0%	0.0%
Negative	0	0
Negative (%)	0.0%	0.0%
Memory size	1.3 KiB	1.7 KiB



Results: Drift Specific Report – IRIS dataset

- Drift-Specific report:
  - Columns affected type
  - Drift type
  - Magnitude of the drift=0.2



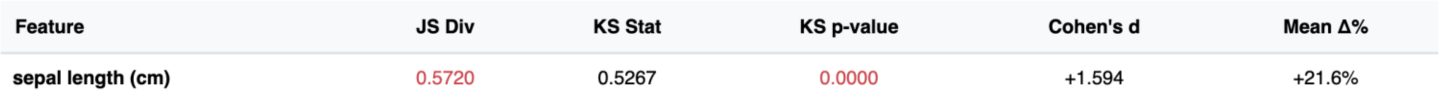
AFFECTED COLUMNS
All numeric

DRIFT TYPE
Abrupt Shift

MAGNITUDE
0.2

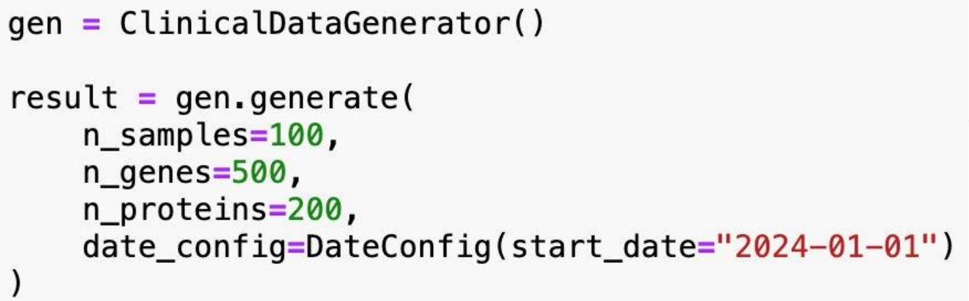
DUPLICATES WITH ORIGINAL
0.0%

- Some statistics about divergence of previous vs new values:
  - JS divergence
  - KS test
  - Cohen's d: Statistic on the difference of means
  - % of the difference of means

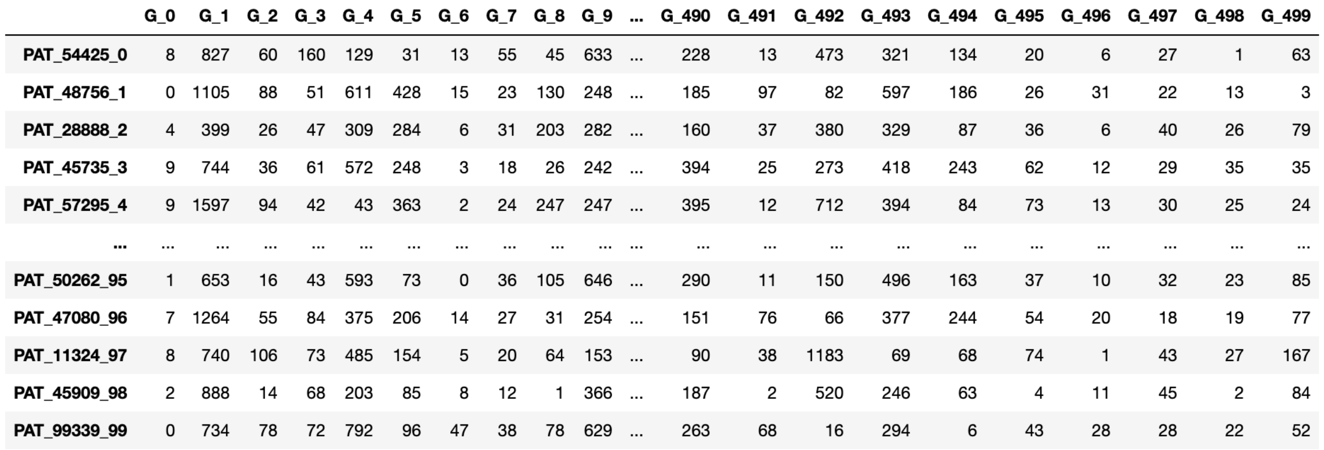


Feature	JS Div	KS Stat	KS p-value	Cohen's d	Mean Δ%
sepal length (cm)	0.5720	0.5267	0.0000	+1.594	+21.6%

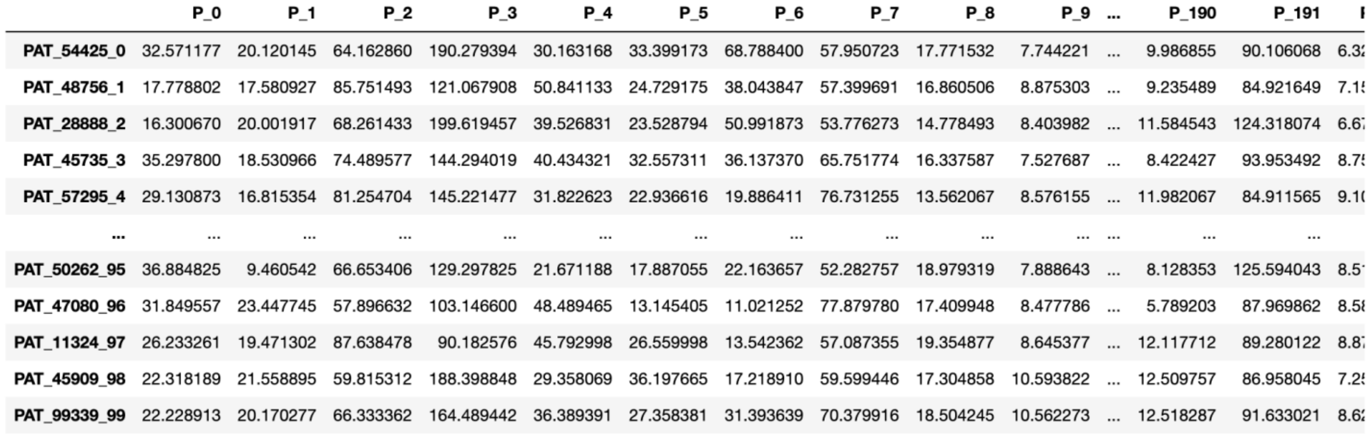
Results: Clinic data generator – Genes+Proteins



```
gen = ClinicalDataGenerator()

result = gen.generate(
    n_samples=100,
    n_genes=500,
    n_proteins=200,
    date_config=DateConfig(start_date="2024-01-01")
)
```

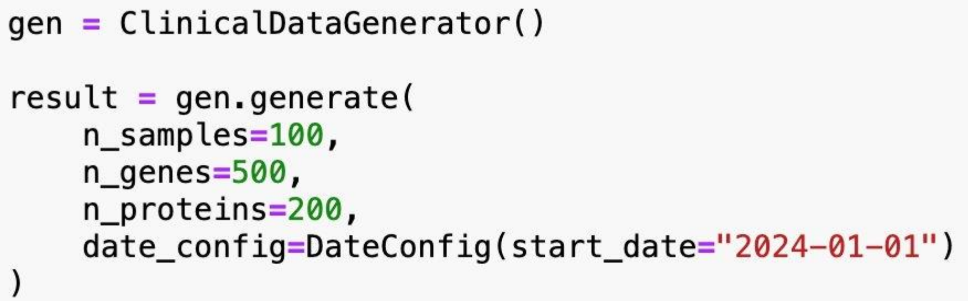


	G_0	G_1	G_2	G_3	G_4	G_5	G_6	G_7	G_8	G_9	...	G_490	G_491	G_492	G_493	G_494	G_495	G_496	G_497	G_498	G_499
PAT_54425_0	8	827	60	160	129	31	13	55	45	633	...	228	13	473	321	134	20	6	27	1	63
PAT_48756_1	0	1105	88	51	611	428	15	23	130	248	...	185	97	82	597	186	26	31	22	13	3
PAT_28888_2	4	399	26	47	309	284	6	31	203	282	...	160	37	380	329	87	36	6	40	26	79
PAT_45735_3	9	744	36	61	572	248	3	18	26	242	...	394	25	273	418	243	62	12	29	35	35
PAT_57295_4	9	1597	94	42	43	363	2	24	247	247	...	395	12	712	394	84	73	13	30	25	24
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
PAT_50262_95	1	653	16	43	593	73	0	36	105	646	...	290	11	150	496	163	37	10	32	23	85
PAT_47080_96	7	1264	55	84	375	206	14	27	31	254	...	151	76	66	377	244	54	20	18	19	77
PAT_11324_97	8	740	106	73	485	154	5	20	64	153	...	90	38	1183	69	68	74	1	43	27	167
PAT_45909_98	2	888	14	68	203	85	8	12	1	366	...	187	2	520	246	63	4	11	45	2	84
PAT_99339_99	0	734	78	72	792	96	47	38	78	629	...	263	68	16	294	6	43	28	28	22	52

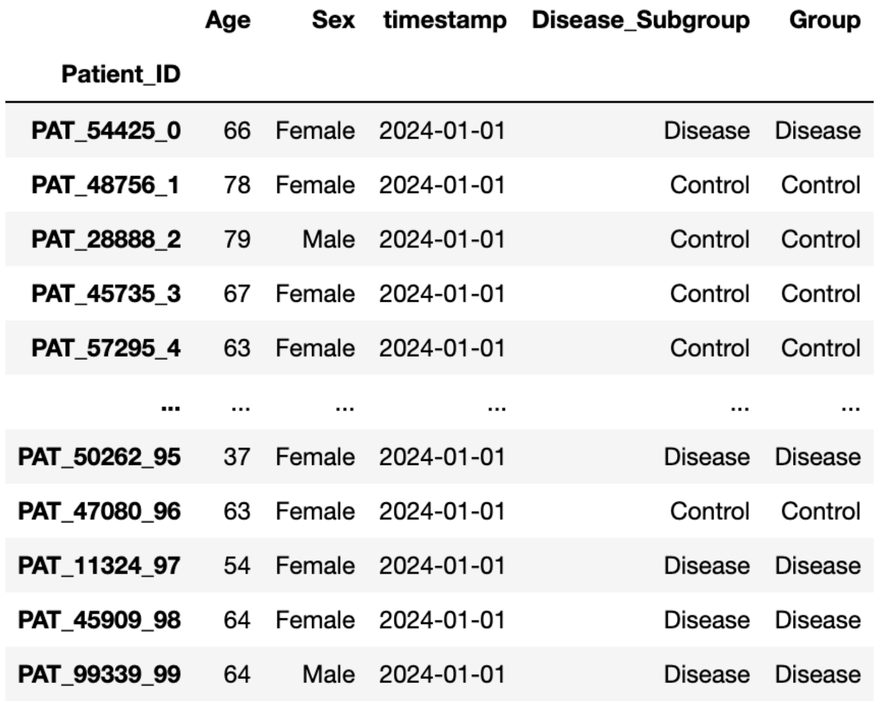


	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	...	P_190	P_191	P
PAT_54425_0	32.571177	20.120145	64.162860	190.279394	30.163168	33.399173	68.788400	57.950723	17.771532	7.744221	...	9.986855	90.106068	6.3
PAT_48756_1	17.778802	17.580927	85.751493	121.067908	50.841133	24.729175	38.043847	57.399691	16.860506	8.875303	...	9.235489	84.921649	7.1
PAT_28888_2	16.300670	20.001917	68.261433	199.619457	39.526831	23.528794	50.991873	53.776273	14.778493	8.403982	...	11.584543	124.318074	6.6
PAT_45735_3	35.297800	18.530966	74.489577	144.294019	40.434321	32.557311	36.137370	65.751774	16.337587	7.527687	...	8.422427	93.953492	8.7
PAT_57295_4	29.130873	16.815354	81.254704	145.221477	31.822623	22.936616	19.886411	76.731255	13.562067	8.576155	...	11.982067	84.911565	9.1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
PAT_50262_95	36.884825	9.460542	66.653406	129.297825	21.671188	17.887055	22.163657	52.282757	18.979319	7.888643	...	8.128353	125.594043	8.5
PAT_47080_96	31.849557	23.447745	57.896632	103.146600	48.489465	13.145405	11.021252	77.879780	17.409948	8.477786	...	5.789203	87.969862	8.5
PAT_11324_97	26.233261	19.471302	87.638478	90.182576	45.792998	26.559998	13.542362	57.087355	19.354877	8.645377	...	12.117712	89.280122	8.8
PAT_45909_98	22.318189	21.558895	59.815312	188.398848	29.358069	36.197665	17.218910	59.599446	17.304858	10.593822	...	12.509757	86.958045	7.2
PAT_99339_99	22.228913	20.170277	66.333362	164.489442	36.389391	27.358381	31.393639	70.379916	18.504245	10.562273	...	12.518287	91.633021	8.6

Results: Clinic data generator – Genes+Proteins



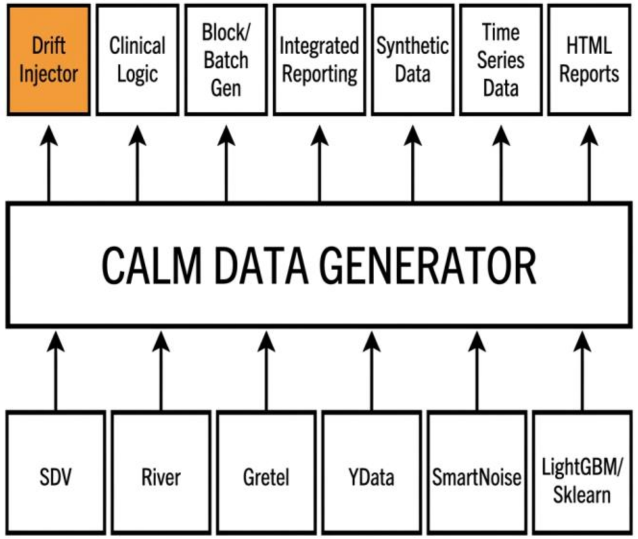
```
gen = ClinicalDataGenerator()

result = gen.generate(
    n_samples=100,
    n_genes=500,
    n_proteins=200,
    date_config=DateConfig(start_date="2024-01-01")
)
```



	Age	Sex	timestamp	Disease_Subgroup	Group
Patient_ID					
PAT_54425_0	66	Female	2024-01-01	Disease	Disease
PAT_48756_1	78	Female	2024-01-01	Control	Control
PAT_28888_2	79	Male	2024-01-01	Control	Control
PAT_45735_3	67	Female	2024-01-01	Control	Control
PAT_57295_4	63	Female	2024-01-01	Control	Control
...	...	...	...	...	...
PAT_50262_95	37	Female	2024-01-01	Disease	Disease
PAT_47080_96	63	Female	2024-01-01	Control	Control
PAT_11324_97	54	Female	2024-01-01	Disease	Disease
PAT_45909_98	64	Female	2024-01-01	Disease	Disease
PAT_99339_99	64	Male	2024-01-01	Disease	Disease

To do

- **Robust testing**
- **Bug fixing**: Missing values, extreme imbalanced, small samples…
- **Add more examples in the documentation**
- **Paper**



Thank you! :-)