DataComp Documentation

Release 0.0.7-dev

Colin Birkenbihl

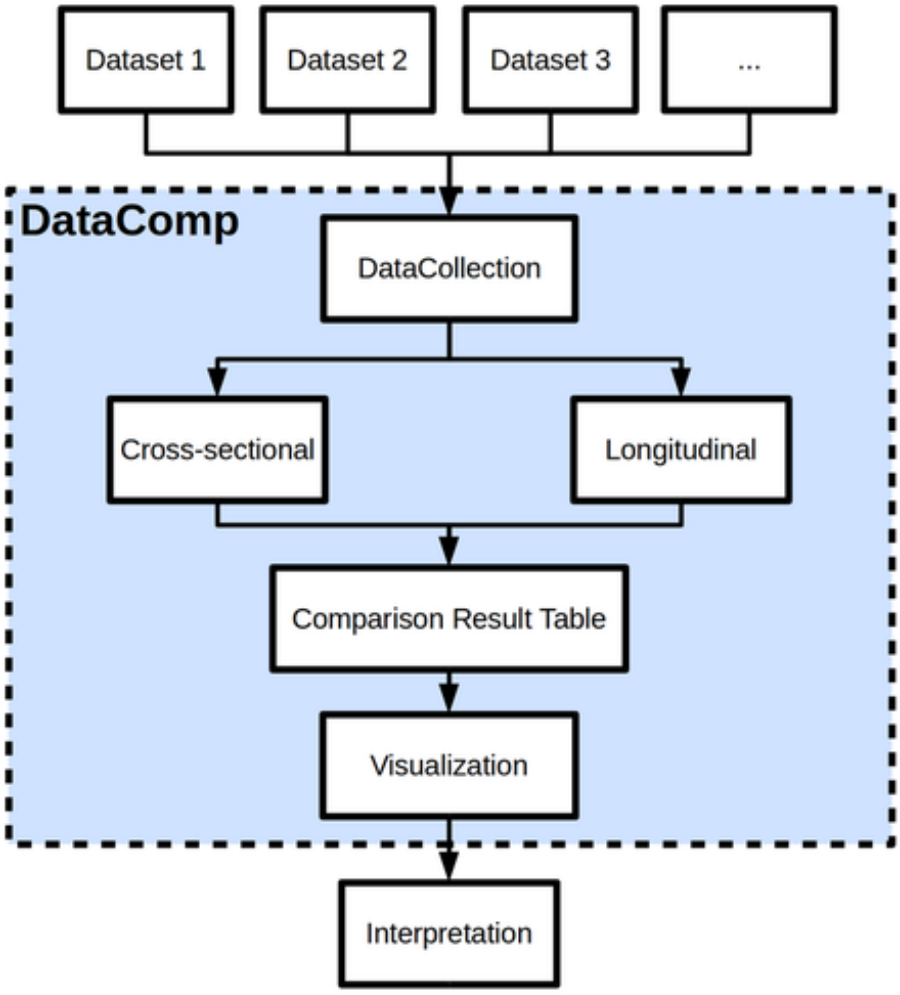
Jun 20, 2019

Contents:

DataComp is an open source Python package for domain independent multimodal longitudinal dataset comparisons. It serves as an investigative toolbox to assess differences between multiple datasets on feature level. DataComp empowers data analysts to identify significantly different and not significantly difference between datasets and thereby is helpful to identify comparable dataset combinations.

Typical application scenarios are:

- Identifying comparable datasets that can be used in machine learning approaches as training and independent test data
- Evaluate if, how and where simulated or synthetic datasets deviate from real world data
- Assess (systematic) differences across multiple datasets (for example multiple sampling sites)
- Conducting multiple statistical comparisons
- Comparative visualizations



This figure depicts a typical DataComp workflow.

CHAPTER 1

Main Features

DataComp supports:

- Evaluating and visualizing the overlap in features across datasets
- Parametric and nonparametric statistical hypothesis testing to compare feature value distributions
- Creating comparative plots of feature value distributions
- Normalizing time series data to baseline and statistically comparing the progression of features over time
- Comparative visualization of feature progression over time
- Hierarchical clustering of the entities in the data sets to evaluate if dataset membership labels are evenly distributed across clusters or assigned to distinct clusters
- Performing a MANOVA to assess the influence of features onto the dataset membership

For examples see *Application examples*.

CHAPTER 2

Links

- Versioning on GitHub
- Documentation on Read the docs
- Distribution via PyPi

2.1 Installation

2.1.1 Using pip

```
python3 -m pip install datacomp
```

2.1.2 From Source

Install the latest version from GitHub.

```
python3 -m pip install git+https://github.com/Cojabi/DataComp.git
```

In editable mode:

This makes sure that the newest version of the code is used when the code is changed locally.

```
git clone https://github.com/Cojabi/DataComp.git
cd DataComp
python3 -m pip install -e .
```

2.2 Application examples

Example notebooks showcasing simple Datacomp workflows and results on simulated data can be found at *DataComp_Examples*:. A reference to a more thorough application example on real world data will be provided in the future. Manuscript in preparation.

- Cross-sectional Comparison Example

In this example a cross-sectional comparison is performed in which only data at one point in time (here at baseline) is considered.

- Longitudinal Comparison Example

The longitudinal examples shows a comparison of the data distribution at each time point in a intra-entity normalized and unnormalized fashion.

2.3 Prerequisites

To ensure proper DataComp functionality some basic assumptions have to be met by the datasets:

- Tabular data formats need to be used (.csv, .tsv, excel etc.)
- Data points / entities are represented as rows
- Features / variables should be columns
- Feature names should be equivalent among the datasets - semantically equal features should bear the same name. Features that are named differently will be treated as features that are not in common.
- Feature values should be represented in the same way (e.g. same variable coding, same categories for discrete variables)
- Any data alternations (e.g. normalization) should be carried out the same way on the features to be compared. Otherwise they will influence comparison results

2.3.1 Common Errors

Make sure that numeric feature columns hold solely numeric data and/or missing values (nan's). String values like for example ">90" must be converted into numerical values other wise errors will occur.

2.4 Interpretation of Results

2.4.1 Statistical Comparison Result-Table

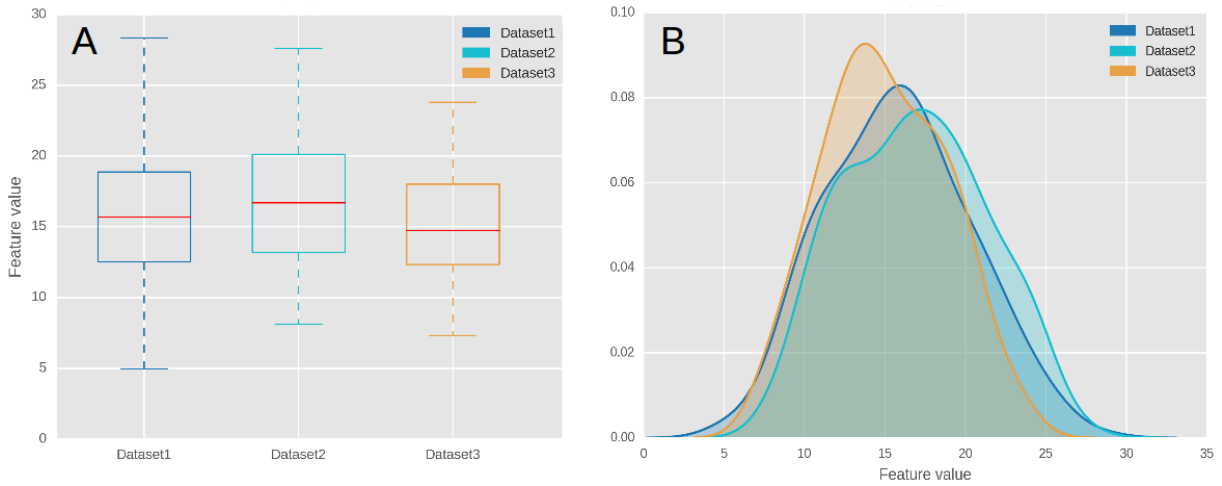
Cross-sectional Comparison

features	datasets	pv	cor_pv	signf	Dataset1	Dataset2	Dataset3
BATCH1	(1, 2)	7.764258e-01	8.855486e-01	False	240	240	198
	(1, 3)	2.544379e-70	0.000000e+00	True	240	240	198
	(2, 3)	1.532594e-72	0.000000e+00	True	240	240	198
BATCH2	(1, 2)	8.995161e-02	4.830466e-01	False	240	240	198
	(1, 3)	3.605956e-52	0.000000e+00	True	240	240	198
	(2, 3)	6.866497e-57	0.000000e+00	True	240	240	198

The index (rows) of the result table consists of the feature name on the outer level and the dataset combination used in the respective comparison as second level index. FEATURE1 (1, 2) hereby declares that this row represents the comparison of FEATURE1 between dataset 1 and 2,where 1 and 2 point to the position of the respective dataset in the DataCollection.

The result table of a feature comparison lists in the columns:

- The p-values
- The familywise error corrected p-values
- A boolean indicator if the test result was significant (a significance level alpha of 0.05 is assumed)
- One column per dataset listing the number of observations available for the comparison of that feature

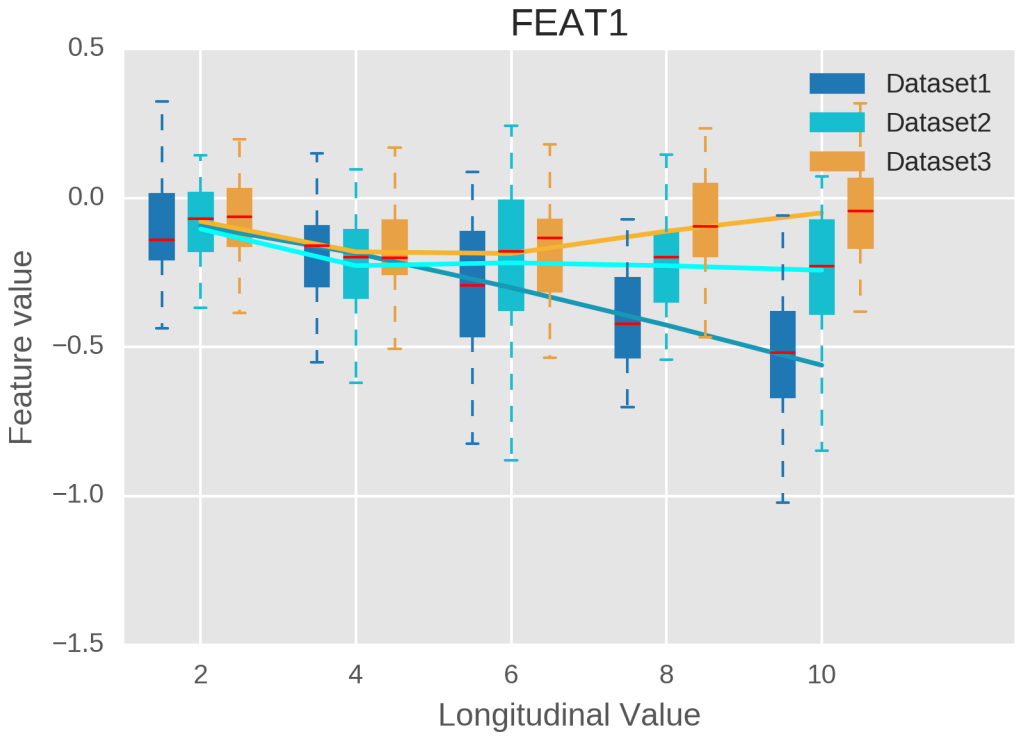


The above figure shows the visualization of data as a boxplot (A) and a comparative kernel density estimate plot (B).

Longitudinal Comparison

			pv	cor_pv	signf	Dataset1	Dataset2	Dataset3
features		datasets						
FEAT1	2	(1, 2)	8.587101e-01	9.949895e-01	False	40	40	33
		(1, 3)	8.288823e-01	9.949895e-01	False	40	40	33
		(2, 3)	7.021697e-01	9.921318e-01	False	40	40	33
	4	(1, 2)	2.118956e-02	1.753164e-01	False	40	40	33
		(1, 3)	1.480527e-01	6.742464e-01	False	40	40	33
		(2, 3)	4.884718e-01	8.673395e-01	False	40	40	33
	6	(1, 2)	7.987259e-01	9.996697e-01	False	40	40	33
		(1, 3)	9.867351e-01	9.996697e-01	False	40	40	33
		(2, 3)	8.548892e-01	9.996697e-01	False	40	40	33
	8	(1, 2)	3.467447e-02	1.617589e-01	False	40	40	33
		(1, 3)	6.807820e-04	5.433296e-03	True	40	40	33
		(2, 3)	4.085755e-02	1.617589e-01	False	40	40	33

Like for the cross-sectional comparison the table is indexed using multiple layers. The first and third index refer again to the feature name and the datasets being compared, respectively. The second layer gives information on the longitudinal dimension. It lists the longitudinal value (e.g. a time point) at which the corresponding statistical tests have been performed.



This figure depicts a visualization of the progression of a feature over multiple longitudinal steps.

2.4.2 Clustering

The confusion matrix of the hierarchical clustering lists the datasets as rows and clusters as columns. Each cell represents the number of entities of that dataset being assigned to the respective cluster. Based on the contingency matrix a chi^2 test is performed to assess if the distribution across clusters is significantlydifferent for the datasets. If so, there might be systematic batch effects present.

2.5 Missing Data

DataComp will handle missing data in several ways if encountered. The amount of data for each variable in each dataset will be displayed next to the hypothesis test results to evaluate reliability.

2.5.1 Dataset Composition

If a single dataframe is the basis for creating a DataCollection it will be split into multiple dataframes based on the values in the column specified by the user to split upon. In the case that one feature contains only missing values (nan's) this feature will be excluded from the corresponding dataframe.

2.5.2 Statistical Comparison

Numerical Features

In the direct comparison of numerical feature distributions missing values (nan's) will not be taken into account. (e.g. distribution [1, 2, nan, 3, 3] will be handled as [1, 2, 3, 3])

Categorical Features

If categories are not present at all in one of the samples, they will be added and their observation count will be set to 0. This can be a problem, since a chi^2-test assumes that there should be at least 5 observations per field in the contingency table. If only one cell is below 5 the assumptions can still be somewhat met but results should be treated carefully and be checked upon. DataComp will through a warning when this is encountered.

2.5.3 Longitudinal Entity Drop-out

For the longitudinal comparison it makes sense to visualize the data availability for each dataset per time point. This can be done using:

```
plot_entities_per_timepoint(datacol, time_col, label_name)
```

An example can be seen here.

For the comparisons at each single time point, missing data is handled as described in the section 'Statistical Comparison'.

2.6 DataCollection

2.7 Statistics

2.8 Visualization

2.9 Utilities

2.10 Longitudinal

2.11 Propensity Score Matching

CHAPTER 3

Indices and tables

- genindex
- modindex
- search