




cuallee: A Python package for data quality checks across multiple DataFrame APIs

Herminio Vazquez ^{1*} and Virginie Grosboillot ^{2*}

1 Independent Researcher, Mexico 2 Swiss Federal Institute of Technology (ETH) * These authors contributed equally.

DOI: [10.21105/joss.06684](https://doi.org/10.21105/joss.06684)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Mehmet Hakan Satman](#) 

Reviewers:

- [@devarops](#)
- [@FlorianK13](#)

Submitted: 10 March 2024

Published: 23 June 2024

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Summary

In today's world, where vast amounts of data are generated and collected daily, and where data heavily influence business, political, and societal decisions, it is crucial to evaluate the quality of the data used for analysis, decision-making, and reporting. This involves understanding how reliable and trustworthy the data are. To address this need, we have created `cuallee`, a Python package for assessing data quality. `cuallee` is designed to be dataframe-agnostic, offering an intuitive and user-friendly API for describing checks across the most popular dataframe implementations such as PySpark, Pandas, Snowpark, Polars, DuckDB, and BigQuery. Currently, `cuallee` offers over 50 checks to help users evaluate the quality of their data.

Statement of need

For data engineers and data scientists, maintaining a consistent workflow involves operating in hybrid environments, where they develop locally before transitioning data pipelines and analyses to cloud-based environments. Whilst working in local environments typically allows them to fit data sets in memory, moving workloads to cloud environments involve operating with full scale data that requires a different computing framework ([Schelter et al., 2018](#)), i.e. distributed computing, parallelization, and horizontal scaling. `cuallee` accomodates the testing activities required by this shift in computing frameworks, in both local and remote environments, without the need to rewrite test scenarios or employ different testing approaches for assessing various quality dimensions of the data ([Fadlallah et al., 2023b](#)).

An additional argument is related to the rapid evolution of the data ecosystem ([Fadlallah et al., 2023a](#)). Organizations and data teams are constantly seeking ways to improve, whether through cost-effective solutions or by integrating new capabilities into their data operations. However, this pursuit presents new challenges when migrating workloads from one technology to another. As information technology and data strategies become more resilient against vendor lock-ins, they turn to technologies that enable seamless operation across platforms, avoiding the chaos of fully re-implementing data products. In essence, with `cuallee` no data testing strategy needs to be rewritten or reformulated due to platform changes.

One last argument in favor of using a quality tool such as `cuallee` is the need to integrate quality procedures into the early stages of data product development. Whether in industry or academia, there is often a tendency to prioritize functional aspects over quality, leading to less time being dedicated to quality activities. By providing a clear, easy-to-use, and adaptable programming interface for data quality, teams can incorporate quality into their development process, promoting a proactive approach of building quality in rather than relying solely on testing to ensure quality.

Data Quality Frameworks

Data platforms have diversified from file systems and relational databases, to full ecosystems including the concept of data lakes (Dumke et al., 2020). Modern platforms host a variety of data formats, beyond traditional tabular data, including semi-structured like JSON (Pezoa et al., 2016) or unstructured like audio or images.

Operating with modern data platforms, requires a versatile data processing framework capable to handle structured and unstructured data, supports data operations in various programming languages, fulfills the imperative and declarative form to data operations from practitioners and does it reliably for any size of data. Apache Spark (Armbrust et al., 2015) represents an exemplar framework due to the wide range of data processing capabilities—batch processing, real-time streaming, machine learning, and graph processing—within a unified framework commended and adopted (O'Reilly Media, 2023) by the data industry.

cuallee is powered by native data engines, including Apache Spark, and offers a robust structure that can be extended to new engines with fully open-source implementation guidelines and rigorous testing. pydeequ (Schelter et al., 2018) is a pioneer in large-scale data quality frameworks and is fully open-source. However, its adoption is limited due to the smaller community of developers proficient in the scala programming language.

On the other hand, great-expectations (Gong et al., n.d.) and soda (Soda Core, n.d.) additionally to an open-source platform also offer commercial options that require registration and issuing of keys for cloud reporting capabilities.





cuallee provides a fully open-source data quality framework designed for both academia and industry practitioners, offering unparalleled performance compared to the aforementioned alternatives.

Performance Benchmark

A reproducible performance benchmark is available in the code repository (Vazquez, 2024). It consists of 38 checks over an open sourced data set (New York City Taxi and Limousine Commission, 2024) made of 19.8 million rows. The validation performs 19 checks for **completeness** and 19 checks for **uniqueness** for each column of the dataset.

The following table (Table 1) provides a summary of the performance comparison:

Table 1: Performance comparison on popular data quality frameworks

Framework	Definitions	Time
great_expectations==0.18.13	python	 66s
soda==1.4.10	yaml	 43s
pydeequ==1.3.0	python	 11s
cuallee==0.10.3	python	 7s

Methods

cuallee employs a heuristic-based approach to define quality rules for each dataset. This prevents the inadvertent duplication of quality predicates, thus reducing the likelihood of human error in defining rules with identical predicates. Several studies have been conducted on the efficiency of these rules, including auto-validation and auto-definition using profilers (Tu et al., 2023).

Checks

In *cuallee*, checks serve as the fundamental concept. These checks (Table 2) are implemented by *rules*, which specify *quality predicates*. These predicates, when aggregated, form the criteria used to evaluate the quality of a dataset. Efforts to establish a universal quality metric (Pleimling et al., 2022) typically involve using statistics and combining dimensions to derive a single reference value that encapsulates overall quality attributes.

Table 2: List and description of the currently available checks

Check	Description	Data Type
is_complete	Zero nulls	<i>agnostic</i>
is_unique	Zero duplicates	<i>agnostic</i>
is_primary_key	Zero duplicates	<i>agnostic</i>
are_complete	Zero nulls on group of columns	<i>agnostic</i>
are_unique	Composite primary key check	<i>agnostic</i>
is_composite_key	Zero duplicates on multiple columns	<i>agnostic</i>
is_greater_than	col > x	<i>numeric</i>
is_positive	col > 0	<i>numeric</i>
is_negative	col < 0	<i>numeric</i>
is_greater_or_equal_than	col >= x	<i>numeric</i>
is_less_than	col < x	<i>numeric</i>
is_less_or_equal_than	col <= x	<i>numeric</i>
is_equal_than	col == x	<i>numeric</i>
is_contained_in	col in [a, b, c, ...]	<i>agnostic</i>
is_in	Alias of is_contained_in	<i>agnostic</i>
not_contained_in	col not in [a, b, c, ...]	<i>agnostic</i>
not_in	Alias of not_contained_in	<i>agnostic</i>
is_between	a <= col <= b	<i>numeric, date</i>
has_pattern	Matching a pattern defined as a regex	<i>string</i>
is_legit	String not null & not empty $^{\backslash}S^{\$}$	<i>string</i>
has_min	min(col) == x	<i>numeric</i>
has_max	max(col) == x	<i>numeric</i>
has_std	$\sigma(\text{col}) == x$	<i>numeric</i>
has_mean	$\mu(\text{col}) == x$	<i>numeric</i>
has_sum	$\Sigma(\text{col}) == x$	<i>numeric</i>
has_percentile	$\%(\text{col}) == x$	<i>numeric</i>
has_cardinality	count(distinct(col)) == x	<i>agnostic</i>
has_max_by	A utility predicate for max(col_a) == x for max(col_b)	<i>agnostic</i>
has_min_by	A utility predicate for min(col_a) == x for min(col_b)	<i>agnostic</i>
has_correlation	Finds correlation between 0..1 on corr(col_a, col_b)	<i>numeric</i>
has_entropy	Calculates the entropy of a column entropy(col) == x for classification problems	<i>numeric</i>
is_inside_iqr	Verifies column values reside inside limits of interquartile range Q1 <= col <= Q3 used on anomalies.	<i>numeric</i>
is_in_millions	col >= 1e6	<i>numeric</i>
is_in_billions	col >= 1e9	<i>numeric</i>
is_t_minus_1	For date fields confirms 1 day ago t-1	<i>date</i>
is_t_minus_2	For date fields confirms 2 days ago t-2	<i>date</i>

Check	Description	Data Type
is_t_minus_3	For date fields confirms 3 days ago t-3	date
is_t_minus_n	For date fields confirms n days ago t-n	date
is_today	For date fields confirms day is current date t-0	date
is_yesterday	For date fields confirms 1 day ago t-1	date
is_on_weekday	For date fields confirms day is between Mon-Fri	date
is_on_weekend	For date fields confirms day is between Sat-Sun	date
is_on_monday	For date fields confirms day is Mon	date
is_on_tuesday	For date fields confirms day is Tue	date
is_on_wednesday	For date fields confirms day is Wed	date
is_on_thursday	For date fields confirms day is Thu	date
is_on_friday	For date fields confirms day is Fri	date
is_on_saturday	For date fields confirms day is Sat	date
is_on_sunday	For date fields confirms day is Sun	date
is_on_schedule	For date fields confirms time windows i.e. 9:00 - 17:00	timestamp
is_daily	Can verify daily continuity on date fields by default. [2,3,4,5,6] which represents Mon-Fri in PySpark. However new schedules can be used for custom date continuity	date
has_workflow	Adjacency matrix validation on 3-column graph, based on group, event, order columns.	agnostic
satisfies	An open SQL expression builder to construct custom checks	agnostic
validate	The ultimate transformation of a check with a dataframe input for validation	agnostic
iso.iso_4217	currency compliant ccy	string
iso.iso_3166	country compliant country	string
Control.completeness	Zero nulls all columns	agnostic
Control.percentage_fill	% rows not empty	agnostic
Control.percentage_empty	% rows empty	agnostic

References

- Armbrust, M., Xin, R. S., Lian, C., Huai, Y., Liu, D., Bradley, J. K., Meng, X., Kaftan, T., Franklin, M. J., Ghodsi, A., & Zaharia, M. (2015). Spark SQL: Relational data processing in Spark. *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, 1383–1394. <https://doi.org/10.1145/2723372.2742797>
- Dumke, A. R., Parchmann, A., Schmid, S., & Hauswirth, M. (2020). Toward data lakes as central building blocks for data management and analysis. *Frontiers in Big Data*, 3, 564115. <https://doi.org/10.3389/fdata.2022.945720>
- Fadlallah, H., Kilany, R., Dhayne, H., El Haddad, R., Haque, R., Taher, Y., & Jaber, A. (2023a). BIGQA: Declarative big data quality assessment. *Journal of Data and Information Quality*, 15. <https://doi.org/10.1145/3603706>
- Fadlallah, H., Kilany, R., Dhayne, H., El Haddad, R., Haque, R., Taher, Y., & Jaber, A. (2023b). Context-aware big data quality assessment: A scoping review. *Journal of Data*

- and *Information Quality*, 15. <https://doi.org/10.1145/3603707>
- Gong, A., Campbell, J., & Great Expectations. (n.d.). *Great Expectations*. https://github.com/great-expectations/great_expectations
- New York City Taxi and Limousine Commission. (2024). *TLC trip record data*. <https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page>
- O'Reilly Media. (2023). *Technology trends for 2023*. <https://www.oreilly.com/radar/technology-trends-for-2023/>
- Pezoa, F., Reutter, J., Suarez, F., Ugarte, M., & Vrgoč, D. (2016). *Foundations of JSON schema*. 263–273. <https://doi.org/10.1145/2872427.2883029>
- Pleimling, X., Shah, V., & Lourentzou, I. (2022). [Data] quality lies in the eyes of the beholder. *Proceedings of the 15th International Conference on PErvasive Technologies Related to Assistive Environments*, 118–124. <https://doi.org/10.1145/3529190.3529222>
- Schelter, S., Lange, D., Schmidt, P., Celikel, M., Biessmann, F., & Grafberger, A. (2018). Automating large-scale data quality verification. *Proc. VLDB Endow.*, 11(12), 1781–1794. <https://doi.org/10.14778/3229863.3229867>
- Soda core. (n.d.). <https://github.com/sodadata/soda-core>
- Tu, D., He, Y., Cui, W., Ge, S., Zhang, H., Han, S., Zhang, D., & Chaudhuri, S. (2023). Auto-validate by-history: Auto-program data quality constraints to validate recurring data pipelines. *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 4991–5003. <https://doi.org/10.1145/3580305.3599776>
- Vazquez, H. (2024). *Cuallee: Performance tests*. <https://github.com/canimus/cuallee/tree/main/test/performance>